Università degli Studi "Roma Tre"
Dipartimento di Matematica e Fisica

Scuola Dottorale in Scienze Matematiche e Fisiche

Sezione di Matematica
XXVI Ciclo

Ph.D. Thesis

# Security and Privacy for Distributed and Resource-Constrained Systems

*Candidate:*
Stefano Guarino

*Advisor:*
Dr. Roberto Di Pietro

*Coordinator:*
Prof. Luigi Chierchia

A.A. 2013-2014

# Abstract

The concept of data security is traditionally associated with cryptography: we are taught to use cryptographic algorithms to protect our digital data, similarly to how we use locks to safeguard our real-life goods. Indeed, in most application settings, properly designed cryptographic tools allow to: (i) protect private information from unauthorized access, (ii) conceal confidential communications to everyone but the intended recipients, and (iii) prove/verify data integrity and authenticity.

In recent times, however, innovative paradigms were introduced for information collection, processing, and storage, that expose data to novel and challenging threats. The proposed models become more and more pervasive in the society, relying on emerging technologies to address an increasing number of scenarios and problems, and to design new solutions for data handling. It is extremely common nowadays to rely on automated information systems and services for: military operations, environmental and industrial process monitoring, health care, alarm systems, remote data storage and elaboration. Many of these application settings impose to implement highly distributed functionalities, to deal with untrusted third parties, and to rely on resource-constrained and physically exposed devices. Traditional security mechanisms are not only hardly feasible, but often even unable to provide the intended protection. Nevertheless, beneficiaries of such services demand reliability, security and efficiency to be comparable to, or better than, standard solutions. To provide a suitable trade-off among these three requirements is the fundamental purpose of most of contemporary research work.

There are two main scenarios that recently attracted the attention of the research community: automated sensing systems, and cloud storage and computing. The application settings are innumerable and of primary importance. The former provide low cost solutions to a variety of military and civilian applications, from home to industrial automation, from harbor to border protection, from health care to wildlife monitoring. The latter permit to enhance reliability without incurring in excessive maintenance costs, and allow big companies to store and process big data in spite of the lack of the necessary infrastructures, as well as private users to securely access and share their data only relying on handheld devices.

In this thesis, we address some aspects of the aforementioned information systems which straddle security and performance issues. We describe innovative techniques able to efficiently provide, at once, service reliability, and data integrity, confidentiality and privacy. First, we propose a secure and efficient data handling scheme for Distributed Sensor Networks (DSNs), where local data sharing is combined with a clever usage of sensors mobility to diffuse information in the most favorable way. Then, we introduce data-access time as one of the main problematics of Cloud Storage systems, and show how a detailed analysis of the components of the system allows to efficiently verify the performances of the storage medium used at the server side, in conjunction with data integrity. Both solutions proposed clearly point the way for future research in the area, highlighting the importance of distributed error/erasure correcting codes for data security in DSNs, and of ensuring other aspects of service reliability, other than just data integrity, in cloud storage applications.

# Ringraziamenti

Le occasioni per ringraziare come meriterebbero le persone che ci circondano non sembrano mai abbastanza. Mentre è semplice dire "grazie" per un piccolo gesto, magari insignificante, è spesso incredibilmente complicato trovare il modo e lo strumento giusto per ringraziare chi quotidianamente ci supporta e ci accompagna, condividendo con noi gioie e dispiaceri.

Il percorso, personale oltre che lavorativo, che ho imboccato poco più di tre anni fa mi ha portato ad essere oggi una persona molto diversa da quella che ero. Ci tengo ad approfittare dell'opportunità rappresentata da questa tesi per ricordare a tutti coloro che, in un modo o nell'altro, mi sono stati accanto in questa faticosa e stimolante esperienza quanto siano (stati) importanti. Senza le persone che sto per citare, difficilmente sarei riuscito ad affrontare nel modo giusto le difficoltà incontrate, e a sfruttare a pieno le opportunità che si sono presentate nel frattempo. Per questo, le ringrazio dal profondo del cuore.

Il primo pensiero va a lei che, già da molto più di tre anni, ha pervaso la mia vita sotto ogni aspetto, impersonificandone dolori, gioie, difficoltà, e speranze più di chiunque altro. É il fine e la ragione di ogni decisione e di ogni passo, e senza lei accanto vagherei senza una meta o una direzione.

*Our love was lost,*
*but now we found it...*

Il ringraziamento alla famiglia, passaggio scontato in queste occasioni, è quanto mai sentito nel mio caso. Per me è sempre stato un punto di riferimento, un luogo di confronto, una fonte inesauribile di insegnamenti. Ho sempre potuto condividere tutto, ed ho sempre trovato braccia aperte pronte per confortarmi, e menti lucide in grado di ispirarmi. E so quanto rara sia la fortuna che mi è capitata. Di recente la famiglia si è allargata, e anche il piccolo nuovo arrivato ha fatto la sua parte, grazie alla sua inconsapevole capacità di alleviare qualsiasi peso al solo sguardo.

Un grazie sentito va al mio supervisor, che ha avuto la pazienza e la forza di spingermi verso un traguardo che a volte sembrava non volessi raggiungere. Lavorare con me è difficile, e la profonda diversità che ci caratterizza deve aver reso ancora più complesso sopportare le mie carenze e gestire le mie stranezze. Eppure non ha desistito, e mi ha insegnato tantissimo.
Insieme a lui, ringrazio gli altri ricercatori con cui ho avuto l'opportunità di collaborare, il coordinatore del dottorato, e i referee e i membri della mia commissione, per il tempo e l'attenzione dedicati a valutare giudiziosamente il mio lavoro.

Impossibile non menzionare i miei "colleghi", se così vogliamo definirli, con cui il legame umano è stato molto più forte e importante di quello lavorativo. Grazie a loro, ho potuto lavorare in serenità ed allegria, imparando molto più di quanto libri e articoli possano insegnare. E per colpa loro, nessun altro ufficio in cui capiterò in futuro sarà mai all'altezza delle aspettative.

Alla mia seconda famiglia, a quelle persone che la vita mi ha dato la fortuna di incontrare lungo la strada e che ormai da tanti anni mi accompagnano, va un ringraziamento speciale. Alcuni oggi sono lontani, ma so che a tenerci uniti è una molla molto resistente, in grado di tendersi indefinitamente senza mai spezzarsi, e di riportarci vicino come un tempo nonappena il destino gliene darà l'occasione. Altri sono con me tutti i giorni, sopportano i miei difetti e sono sempre pronti a tendere una mano, e non so come ringraziarli per questo. Spero solo di riuscire a fare per loro almeno parte di quello che loro fanno per me, perchè giuro che ci provo con tutte le mie forze.

Infine, un pensiero va a papà, che probabilmente oggi sarebbe fiero di me più di quanto io meriti in realtà. É grazie ai suoi sforzi e ai suoi sacrifici se la vita mi ha dato possibilità che a lui ha negato, e questo è qualcosa che non dimenticherò mai.

*cosa ci aspetta? mi sa che è meglio non saperlo,*
*adesso attento a ciò che vuoi perché potresti anche ottenerlo...*

# Contents

# List of Figures

# List of Tables

# Introduction

The work presented in this thesis is organized into two major parts: in Part I, we discuss secure data handling in Distributed Sensor Networks (DSNs), while in Part II we address privacy and service integrity in Cloud Storage (CS) applications. The two considered scenarios, DSNs and CS, are very different from each other, the former being a model for automated environmental monitoring, and the latter being a paradigm for remote storage services. However, the characteristics of DSNs and CS make the analysis of security, privacy and efficiency concerns in the two contexts somehow similar. In both cases, in fact, many security aspects cannot be addressed with standard techniques, mainly due to data physical exposure, to the limited resources of the devices involved, and to the many hardly predictable variables involved (*e.g.*, the random and ever-changing topology of a DSN, and the performances of providers' hardware and network traffic in CS). The most reasonable approach is to put perfect security aside, and to implement probabilistic protocols, whose reliability strongly depends on a careful analysis of the system model. Information security in these contexts requires a well-balanced combination of technological awareness and mathematical background: while in similar scenarios the design of security mechanisms often consists in a clever combination of cryptographic tools (whose individual security is already well established), in DSNs and CS systems it is necessary to precisely model the application setting, to understand what features can be disregarded because of their negligible impact, and to identify how instead the focus of the analysis depends on all relevant parameters.

Since the thesis is divided into two main segments, we will separately introduce the two scenarios at the beginning of each part, instead of providing a communal portrait here. In both cases, we will describe in detail the application setting, motivate the requirement for novel approaches, and explain why our solution differentiates from past ones and provides a new and remarkable contribution.

In the remainder of this general introduction, let us instead depict the road map of the thesis, and recap the contributions provided by the candidate during his PhD studies.

## Road Map

The thesis is organized upon two main pillars:

- data security in DSNs, discussed in Part I, and
- service reliability in CS systems, discussed in Part II.

Each of these two parts further develops along three chapters. For what concerns Part I:

- Chapter 1 presents an overview of information security in Wireless Sensor Networks (WSNs), the most known paradigm in the literature for automated data sensing;
- Chapter 2 discusses the problematics introduced by Unattended WSNs (UWSNs), Mobile WSNs (MWSNs), and Participatory Sensing (PS), which flow together in the more general model of Distributed Sensor Networks (DSNs), characterized by two main features: lack of a centralized control, and (randomly) mobile sensors;
- Chapter 3 describes in detail the proposed solution for data security in DSNs, based on the concepts of information sharing and diffusion.

Part II consists of the following steps:

- Chapter 4 discusses the main issues of Cloud Storage applications, and the solutions proposed so far;
- Chapter 5 introduces the concept of Provable Storage Medium (PSM), as a complementary aspect of the general problem of Provable Data Possession (PDP), describing what features of the system model are particularly relevant for its purposes;
- Chapter 6 describes the solution we propose to efficiently address, at once, PSM and PDP.

Finally, Chapter 7 concludes the thesis, presenting a brief summary of the contributions provided, and delineating promising and stimulating future lines of research.

## Contributions of the Candidate

The two main parts of this thesis correspond to the two main research fields explored by the candidate during his PhD studies. In the following, we try to summarize the contributions provided in each of them.

**Distributed Sensor Networks** In the field of automated sensing systems, the contributions provided by the candidate can be summed up as follows.

- Most of the content of Chapters 1 and 2 is an extract of the paper [1], recently submitted to the Elsevier Computer Communications. The paper provides a survey of security issues and countermeasures of wireless ad-hoc networks, focusing in particular on five main paradigms: Wireless Sensor Network (WSN), Unattended Wireless Sensor Network (UWSN), Wireless Mesh Network (WMN), Delay Tolerant Network (DTN), and Vehicular Ad-hoc Network (VANET).
- Chapter 3 anticipates an extension of the papers [2] and [3]. To concurrently address data confidentiality and availability in DSNs, in [2] and [3] we propose to share data locally as soon as they are sensed, and to leverage nodes mobility to efficiently diffuse the generated pieces of information. In this extension, we show that the proposed solution can as well enhance the level of protection against fake-data injection and information traceability. Further, we discuss more deeply the impact of the characteristics of the mobility model of the sensors on the viability and effectiveness of the proposed solution.
- The candidate also addressed a slightly different scenario: Unattended Wireless Sensor Networks (UWSNs), and in particular all settings where confidentiality is of minor importance with respect to sensed data availability, and therefore replication is the most reasonable approach. In this case, we propose an analysis of information survivability based on the mathematics of epidemic models, and show how the replication rate can be properly tuned to limit energy consumption without affecting information recovery. The study builds on [4], which sensibly extends and improves.

**Cloud Storage** For what concerns the security of remote storage and computing facilities, the contributions provided by the candidate can be summed up as follows.

- The content of Chapters 5 and 6 composes a substantial work able to introduce, analyze and address the new problem of Provable Storage Medium (PSM) in data storage outsourcing. The work has been recently submitted to the IEEE Transactions on Services Computing, and we are expecting a response soon. In this contribution, we stress that in many cloud storage systems users cannot be simply pleased with Provable Data Possession (PDP), because data access time is as important as data integrity. Consequently, we introduce the concept of PSM, and show how previous PDP solutions can be properly adjusted so as to enable to efficiently check, contextually to data integrity, that data are stored on a storage medium whose performances comply with what stated in the Service Level Agreement (SLA).

- Finally, the candidate worked on an innovative approach to cloud computing applications. Even if the idea can be easily extended to many other types of computations, in this contribution we focus on a privacy preserving scheme for the outsourcing of modular exponentiations by a resource-constrained device. In particular, we show how privacy can be enforced by simply feeding the service provider with a large input, where the real input variable is somehow hidden, but such that the desired result can be easily extracted from the output returned by the provider. The security analysis of the proposed protocol is extremely technical, and based on advanced probabilistic tools.

## List of Research Works of the Candidate

Works accepted for publication:

1. R. Di Pietro, and S. Guarino. Confidentiality and Availability Issues in Mobile Unattended Wireless Sensor Networks. In Proceedings of the 4th IEEE International Workshop on Data Security and PrivAcy in wireless Networks (D-SPAN'13) – In conjunction with IEEE WoWMoM 2013, pages 1–6, 2013. [2]
2. R. Di Pietro, and S. Guarino. Data Confidentiality and Availability Via Secret Sharing and Node Mobility in UWSN. In Proceedings of the 32nd IEEE International Conference on Computer Communications (IEEE INFOCOM'2013), pages 205–209, 2013. [3]

Works submitted and currently undergoing the review process:

3. R. Di Pietro, S. Guarino, N.V. Verde, and J. Domingo-Ferrer. Security in Wireless Ad-Hoc Networks – A Survey. Submitted to the Elsevier Computer Communications. [1]
4. S. Guarino, E.S. Canlar, M. Conti, R. Di Pietro, and A. Solanas. Provable Storage Medium for Data Storage Outsourcing. Submitted to the IEEE Transactions on Services Computing. [5]

Works currently being finalized:

5. R. Di Pietro, and S. Guarino. Data Security in Distributed Sensor Networks? Share Information Locally, and Keep Moving!
6. R. Di Pietro, S. Guarino, and N.V. Verde. Epidemic Data Survivability in Unattended Wireless Sensor Networks.
7. R. Di Pietro, and S. Guarino. Privacy Preserving Exponentiation Outsourcing For Embedded Systems.

Part I

# Data Security in Distributed Sensor Networks

# 1 Security in Wireless Sensor Networks

A wireless network makes use of radio signals to exchange data between two or more physical devices, usually called “nodes” of the network. The lack of wires permits to overcome most limitations of traditional wired networks, allowing deployment in hostile environments or mobile scenarios. When nodes do not depend on any preexisting infrastructure, wireless networks take the name of *wireless ad-hoc networks*. In this case, communications rely on the ability of the nodes to form a multi-hop radio network.

Pervasive mobile and low-end wireless technologies make the wireless scenario exciting and in full transformation. Relying on such emerging technologies, new paradigms were recently introduced to automatically interact with the environment. One of the main challenges currently addressed by the research community consists in designing systems able to model information sensing and elaboration. To reduce deployment and managing costs, it is highly desirable to devise solutions that do not rely on constant centralized control, or that do not even require the deployment of specific networks. However, to fully unleash the potential of similar solutions in the industry and society, there are two pillars that cannot be overlooked: security and privacy. Both properties are especially relevant if we focus on ad-hoc wireless networks, where devices are required to (distributedly) cooperate – *e.g.* from routing to the application layer – to attain their goals.

Among wireless ad-hoc networks, the Wireless Sensor Network (WSN) paradigm plays a predominant role. The nodes of a WSN have sensing capabilities and are appointed to monitor and collect information from the environment. This first chapter presents a survey of emerging and established technologies for WSNs, with particular attention paid to their security and privacy features and deficiencies. In Section 1.1, we present a precise description of the WSN model, summarizing the main vulnerabilities and security requirements imposed by the severe resource constraints and the physical exposure of the sensors. In Section 1.2, we propose a comprehensive classification of known attacks to WSNs, and we delineate the main countermeasures proposed so fare in the literature. It is worth to underline the primary importance of this chapter, because many of the attacks and of the challenges discussed for WSNs represent a threat in most other automated sensing systems as well.

## 1.1 Wireless Sensor Networks

A Wireless Sensor Network (WSN) consists of sensors-equipped nodes, called *motes* or simply *sensors*, sensing the environment and reporting the collected data to one or more trusted gateway nodes, called *sinks*. In traditional scenarios, at least one sink is always on-line and alive, and the sensors collaborate to promptly send all sensed data to such sink. Indeed, to preserve the information even in the presence of sensors malfunctioning or failure, it is preferable to get the data off the sensors as quickly as possible. The persistent presence of the sink is fundamental in many high failure-rate environments (*e.g.*, fire or evacuation systems), where there might be very little time between the detection of an event and the destruction of the sensors that observed it [6]. In similar settings, an adversary has only few chances to erase or to compromise the sensed data before they are received by the sink. Constantly present sinks may also play a coordination role. In general, however, the frequency and impact of the sinks' presence in the network is highly variable according to the setting [7], so motes are often required to self-organize in a distributed way.

WSNs are usually sensibly (sometimes even orders of magnitude) larger than similar ad-hoc networks, and are often deployed in hostile environments and over wide geographic areas. Motes have limited computational power, memory and energy supply, which, together with the adverse working conditions, make them particularly prone to failures. Despite many energy harvesting solutions proposed so far, recharging is still considered hardly feasible, and motes are usually regarded as "disposable" devices. Due to the complexity of replacement and management operations, maximizing lifetime and productivity is of paramount importance. In essence, WSNs are ad-hoc networks with additional and more stringent constraints. They need to be more energy-efficient and scalable than other ad-hoc networks, which exacerbates the security challenges.

Initially, the development of WSNs was mainly motivated by military purposes, but nowadays WSNs are becoming pervasive systems, used in several fields, from home automation to border monitoring. However, military applications, together with automated medical systems, still represent the context where security aspects are more relevant. In both cases, the network handles critical information, hence to ensure data availability is crucial. Further, classified military data and private patients health-status information, raise the concern for confidentiality and privacy.

WSN applications need to contrast most security issues communal to conventional networks, like message injection, eavesdropping, impersonation, etc. However, the design of a security infrastructure in WSNs must pervade any layer of the system, from the application layer to the physical layer (that is often considered secure in conventional settings). Further, mainly because of their limited resources, standard techniques such as tamper-proof hardware, secure routing, public-key cryptography, etc., do not suit WSNs. Specific solutions for WSNs are required, that must be conceived with these low-end devices in mind.

There are two specifications available for WSN communication: IEEE 802.15.4 [8] and ZigBee [9]. The first is a standard for low-rate wireless personal area networks that was developed by IEEE (Institute of Electrical and Electronics Engineers) and contains a number of security suites. Basically, it provides access control, integrity, confidentiality and replay protection; however, it does not deal with authentication or key exchange. IEEE 802.15.4 defines a communication layer at level 2 in the OSI (Open System Interconnection) model and its main purpose is to allow communication between two devices. ZigBee is built upon IEEE 802.15.4. This standard defines a communication layer at level 3 and above in the OSI model. Its main purpose is to create a network topology (hierarchy) to let a number of devices communicate among them, and to add extra communication features such as authentication, encryption and association. The ZigBee network layer natively supports star, tree and generic mesh networks.

### 1.1.1 Vulnerabilities of WSNs

Generally speaking, several vulnerabilities can be identified in WSNs. At a very abstract level they can be related to one of the following issues, which can be more generally associated to most ad-hoc networks:

**Vulnerability of the channel** Messages can be eavesdropped and fake messages can be injected or replayed into the network, without the hurdle of needing physical access to network components.

**Vulnerability of the nodes** Nodes may not be physically protected, and are therefore more prone to capture and tamper attacks. If an adversary gets full access to a node, he can: (i) steal sensitive information, (ii) reprogram the node and change its behavior, or (iii) physically damage hardware to terminate the node. Due to nodes vulnerability, secret keys cannot be simply issued when the network is deployed, but a secure and efficient key management scheme is crucial.[1] The limited communication, storage, and processing capacities of the sensors, as well as their scant energy supply, represent further sources of vulnerabilities that an adversary can conveniently leverage. In fact, several attacks can be designed with the scope of depleting the resources of the nodes, so as to obstruct the normal functioning of the network.

**Absence of infrastructure** Many WSNs are supposed to operate independently of any fixed infrastructure. This makes most classical security solutions, based on certification authorities and on-line servers, inapplicable. Security and privacy *de facto* rely on distributed cooperation among (possibly uncooperative) nodes. Uncooperative nodes can be categorized as *faulty, malicious*, and *selfish*. Malicious and selfish nodes deliberately interfere with the normal behavior of the network: while the former aim at disrupting the normal functionalities of the network, the latter try to increase their own revenue, at the expenses of the other nodes.[2] Faulty nodes are simply nodes that do not behave as expected due to some malfunctioning, and can be considered a minor sub-case of the previous ones.

**Dynamically changing topology** The topology of a WSN is potentially ever and quickly changing. Sophisticated routing protocols are often needed, but they may introduce new problems that need to be carefully evaluated. Indeed, incorrect routing information can be generated by compromised nodes, or as a result of some topology changes.[3]

### 1.1.2 Security Requirements of WSNs

Due to their application constraints, WSNs pose more security requirements than traditional networks. In (some) order of importance, such requirements can be listed as follows.

**Availability** The services provided by the network must be always available (often in a timely manner), despite of any malfunctioning of the system. Resource depletion attacks are the main class of attacks aiming at subverting this property. Resistance to such attacks is therefore of primary importance.

**Integrity** Any accidental or malicious alteration to the information stored and exchanged in the network must be (promptly) detected, and possibly thwarted.

[1]The reader can refer to [10] for a survey on key management protocols.
[2]For an overview on the problems introduced by selfishness, the reader can refer to [11].
[3]For a survey of secure routing protocols specific to ad-hoc networks, we suggest [12, 13].

**Confidentiality** Secret information stored and exchanged in the network must not be divulged to unauthorized parties. In some cases, even the existence itself of a communication between two end-points must be hidden. Cryptographic tools are the typical, but not unique, countermeasure to confidentiality threats. In dynamically changing systems, where nodes can join or leave the network, so-called *forward* and *backward secrecy* need to be addressed as well. *Forward secrecy* means to deny access to any future communication to nodes that left the network. Conversely, *backward secrecy* means to ensure that new nodes are not able to access any message sent before they joined the network.

**Privacy** Private (meta)data–not strictly necessary for the network purposes–must be concealed to everyone, including the network authority. Privacy and confidentiality must not be confused: while the latter concerns hiding to outer entities data used by the network to provide the intended services, the former refers to avoiding that the network becomes excessively intrusive, gathering information that it is not allowed to access.

**Authorization** Only authorized nodes must be able to gain access to the network, and only authorized entities must be able to enjoy the services provided by the network.

**Authentication** It must be always possible to verify the identity of the sender of any message exchanged in the network. Unless it is in control of a corrupted node, no attacker should be able to forge a message, though making it indistinguishable from a legitimate message.

**Non-repudiation** To be able to find and separate compromised nodes, it must be impossible for the sender of a message to successfully challenge the authorship of that message.

**Freshness** It must be always possible to verify the newness of data exchanged in the network, to prevent any adversary to re-use old messages to mislead network services.

## 1.2 Classifications of Attacks to WSNs

At a high level, attacks against WSNs can be classified based on the status of the attacker, on its behavior, and on the purpose of the attack.

**Status** The first classification is based on whether the attacker is an *outsider* or an *insider*. Outsider attackers are entities that do not belong to the network but want to disrupt the provided service. Insider attackers are legitimate nodes behaving in a malicious way.

**Behavior** The second classification distinguishes between *passive* and *active* attacks. The former only consist in eavesdropping communications, and monitoring and analyzing the behavior of the network, without interfering with it. The latter include physical access to a portion of the network, and attempts to modify the normal behavior of the network.

**Purpose** The third categorization depends on the purpose of the attack. Attacks on network *availability* and *service integrity*, aim at disrupting the services provided by the network. Many denial-of-service, routing and physical attacks fall within this category. Attacks against *privacy* and *confidentiality* are attacks that try to gain insight on data exchanged in the network and on the network topology. Finally, attacks against *data integrity* try to alter the data that are transmitted. Malicious nodes can inject false messages, modify existing ones, replicate old packets or entire nodes, etc.

In the following, we will provide a more precise categorization of the attacks that can be mounted against WSNs. We will describe in detail several threats, and we will point out the existing countermeasures. Table 1.1 summarizes the attacks that we will take into consideration, their categorization according to the above classifications, and the corresponding countermeasures.

### 1.2.1 Attacks Against Network Availability and Service Integrity

Attacks against network availability and service integrity are often referred to as denial-of-service (DoS) attacks: an adversary attempts to disrupt, subvert or destroy the services provided by the network. DoS attacks can have as a target any layer of the sensor network. Indeed, known attacks perform on the physical, the data link, the network and the transport layers. In this section, we will analyze existing DoS attacks layer by layer.

#### Physical Layer

In WSNs, attacks to the physical layer can target the communication channel or the sensors. In the first case we speak of *jamming* attacks, while in the second of *tampering* attacks.

**Jamming** A jamming attack can be seen as noise created by an attacker with the aim of partially or entirely disrupting a legitimate signal. Such noise is generated using a device called *jammer*, able to interfere with the radio frequencies used by the sensors. The jamming activity is effective only if the signal-to-noise ratio is less than 1. Depending on its transmission power, the jammer may disturb the entire network or a smaller portion of it. If ignored in the initial

| Target | Layer | Attack | Countermeasures | Attacker | | Attack | |
|---|---|---|---|---|---|---|---|
| | | | | **Internal** | **External** | **Active** | **Passive** |
| Network Availability and Service Integrity | Physical | Jamming | Detection techniques, proactive, reactive, and mobile agent-based countermeasures | | × | × | |
| | | Tampering | Tamper-proofing, software tamper detection, sensor monitoring | | × | × | |
| | Link | Collision | Forward error-correcting codes | | × | × | |
| | | Exhaustion | Rate limitation | | × | × | |
| | | Unfairness | Error-correcting codes | × | | × | |
| | | Sleep Deprivation | Anti-replay protection, strong link-layer authentication, and broadcast attack protection | | × | × | |
| | Network & Routing | Routing Information | Authentication, MAC | × | | × | |
| | | Hello Flooding | Authentication, bi-directionality checking, signal strength | | × | × | |
| | | Black Hole | Authentication, REWARD, watchdog and pathrater | | × | × | |
| | | Sink Hole Attack | Authentication, monitoring, secure routing | | × | × | |
| | | Selective Forwarding | Authentication, IDS, multi-hop acknowledgements, multipath routing | | × | × | |
| | | Wormhole Attack | Authentication, packet leashes | | × | × | |
| | | Sybil | Authentication, radio resource testing, key validation for random key pre-distribution, position verification | | × | × | |
| | Transport | Flooding | Client puzzles, cryptographic techniques | × | | × | |
| | | Desynchronization | Authentication | | × | × | |
| Privacy and Secrecy | Physical | Eavesdropping | Cryptographic techniques | | × | | × |
| | Network | Traffic Analysis | Randomized communications | × | | × | |
| Data Integrity | Physical | Node Replication | Emergent properties | | × | × | |
| | Network | Packet Injection | Data authentication | × | × | × | |
| | | Packet Duplication | Data authentication | × | × | × | |
| | | Packet Alteration | Data authentication | × | × | × | |

TABLE 1.1: Attacks against Wireless Sensor Networks

WSN design, a jamming attack can easily disrupt a network, regardless of higher level security mechanisms. Jamming can be classified as follows [14]:

- *Spot* jamming is the simplest jamming technique. The attacker directs all its compromising power against a single frequency. It is usually effective, but it may be avoided by changing the frequency used.
- *Sweep* jamming targets multiple frequencies in quick succession, by rapidly shifting the target frequency. Since the activity of the attacker is not continuous, the effectiveness of this type of attack is limited. However, in WSNs it can force many retransmissions due to packet loss.
- *Barrage* jamming concurrently targets a range of frequencies. However, as the attacked range grows, the output power of jamming is reduced proportionally.
- *Deceptive* jamming consists in fabricating or replaying valid signals on the channel incessantly, thereby occupying the available bandwidth and trying to destroy the network service. It can be applied to a single frequency or a set of frequencies.

First generation sensor nodes used single-frequency radios, and were therefore vulnerable to narrowband noise, whether unintentional or malicious.[4] More recent motes use direct-sequence spread spectrum to reduce vulnerability to noise.[5] More generally, several countermeasures can be used against the various jamming attacks. Frequency-Hopping Spread Spectrum (FHSS), Direct Sequence Spread Spectrum (DSSS), Hybrid FHSS/DSSS, Ultra Wide Band (UWB) technology, antenna polarization, directional transmission, and regulation of the transmission power are a few examples [15–17]. However, they do not defeat an adversary with knowledge of the spreading codes or hopping sequence. Indeed, these are not secret, but either standardized (in IEEE 802.15.4) or derivable from node addresses (in Bluetooth).

Existing security schemes that address jamming attacks in WSNs can be broadly classified as follows:[6]

- *Detection* techniques aim at instantly detecting jamming attacks. As observed in [18], signal strength, carrier sensing time or packet delivery ratio individually are unable to conclusively detect the presence of a jammer. To improve detection, the authors of [18] introduce the notion of consistency checking, where the packet delivery ratio is used to classify a radio link as having poor utility, and then a consistency check is performed to classify whether poor link quality is due to jamming.

[4]*e.g.*, Mica2 and prior motes used the Chipcon CC1000 transceiver, operating at 433 or 900MHz.

[5]*e.g.*, MICAz and Telos motes use the Chipcon CC2420, which operates at 2.45 GHz.

[6]For a comprehensive summary of counteractions against jamming in WSNs, we remand the reader to [14].

- *Proactive* countermeasures make a WSN immune to jamming attacks rather than reactively respond to such incidents. An example is DEEJAM, a protocol proposed to defend against stealthy jammers using IEEE 802.15.4-based hardware [19]. To contrast adversaries that use hardware with same capabilities as the deployed nodes, it uses four defensive mechanisms altogether:
  - *Frame masking* defends against attackers jamming only when their radio captures a multibyte preamble and a Start of Frame Delimiter (SFD) sequence.
  - *Channel hopping* defends against attackers that try to detect radio activity by periodically sampling the Radio Signal Strength Indicator (RSSI), and start jamming when RSSI is above a programmable threshold.
  - *Packet fragmentation* consists in breaking each packet into short fragments, transmitted separately on different channels and with different SFDs. If the transmission frequency changes fast enough, the attacker cannot start jamming on the right frequency in time.
  - *Redundant encoding* tackles an attacker that blindly jams a single channel using short pulses. It allows packet recovery even if a fragment is corrupted, but energy and bandwidth usage are increased.
- *Reactive* countermeasures enable reaction only upon the incident of a jamming attack. A perfect example is the JAM algorithm proposed in [20], which enables the detection and mapping of jammed regions to increase network efficiency. In practice, nodes near the border of a jammed region notify their neighbors, which start mapping the region that is currently jammed by exchanging mapping messages. When the jammer moves or simply stops the attack, the jammed nodes recover and notify this change to their neighbors.
- *Mobile agent-based* countermeasures leverage Mobile Agents (MAs), *i.e.*, autonomous programs that can move from host to host and act on behalf of users towards the completion of an assigned task. An example is the JAID protocol presented in [21], where MAs explore the network incrementally fusing the data as they visit the nodes. Firstly, to identify near-optimal itineraries for the MAs, JAID separates the network into multiple groups of nodes, calculates local near-optimal routes through each group, and assigns these itineraries to individual agent objects. Then, such itineraries are modified using the JAM algorithm, so as to avoid jammed areas, while not harming the efficient data dissemination performed by normally working sensors.

**Tampering** A wide range of active attacks, generally carried out by outsiders, all rely on a communal approach: gaining physical access to a subset of sensors by tampering with their hardware. DoS attacks are only one of the possible ways an adversary can leverage tampering. More generally, the purpose may be to modify the behavior of the nodes, to replace them with malicious sensors under the control of the attacker, or to steal confidential data and cryptographic material [22–24]. To provide a clearer exposition, here we will only discuss resilience to tampering itself. Higher-level countermeasures to attacks for which physical access to the sensors in only a prerequisite will be discussed in the pertaining section. The primary defense against physical tampering focuses on building tamper-resistant sensors [25]. However, although tamper-resistant hardware is becoming cheaper, in most cases it is not a feasible option. As an alternative, softwares were specifically designed to detect tampering attempts, and promptly delete sensitive data (such as cryptographic keys) before executing a self-termination protocol. Tampering with current sensor node hardware has been investigated in [26]. The authors show that attacks that can be executed without interruption of the regular node operation usually have a minor impact. All most serious attacks, which result in full control over a sensor node, require the absence of the node from the network for a substantial amount of time. Therefore, simply monitoring sensor nodes for periods of long inactivity can be considered a good defensive strategy.

### Link Layer

In WSNs, several attacks can be mounted on the data-link layer. All such attacks share two main objectives: (i) depleting the energetic resources of the sensors, relying on the fact that most energy consumption in WSNs is due to communication, and (ii) degrading the timeliness of the service.

**Link Layer Collision** This attack is very similar to jamming in the physical layer. It occurs when an attacker uses his radio to identify the frequency used by the WSN, and, as soon as he hears the start of a legitimate message transmission, he sends a signal for as little as one octet (or byte) in order to corrupt the entire message [27]. The only evidence of the attack is the reception of an incorrect message, which is detected when a link layer frame fails a cyclic redundancy code (CRC) check. In that case, the link layer automatically discards the entire packet, thereby causing energy and bandwidth waste. A possible countermeasure is provided by forward error-correcting codes (FEC), able to reactively recover lost information [28].

**Link Layer Exhaustion** This attack occurs when the attacker manipulates protocol efficiency measures and causes nodes to expend additional energy. Providing a rate limitation by allowing nodes to ignore excessive network requests from a node is an effective countermeasure against this attack.

**Unfairness** In an unfairness attack, the adversary transmits a large number of packets when the medium is free, to prevent honest sensors from transmitting legitimate packets. As a result, the quality of service degrades and real-time deadlines are possibly missed. However, this attack is usually considered a weak form of DoS, because it can be limited by using smaller frames, in such a way that the channel is only captured for a small amount of time.

**Sleep Deprivation Torture** In WSNs, a sleep mechanism is used by the nodes to adjust their operation mode and extend their lifetime. At full power, a sensor can run for approximately two weeks before exhausting its power resources. To the contrary, if nodes remain in sleep mode and activate as little as possible (*e.g.*, around 1% of the time), their batteries can last even more than a year. As the name suggests, the "Sleep Deprivation Torture" or "denial-of-sleep" attack, firstly introduced in [29], aims at preventing a sensor from sleeping. According to how sleeplessness is induced, it can be classified into two categories [30]: (i) *service request power attacks*, which intensively repeat usual service requests, and (ii) *benign power attacks*, which solicit power-intensive operations on the device under attack. In [31], the authors proposed three ways to lessen the effect of these attacks: *strong link-layer authentication*, *anti-replay protection*, and *broadcast attack protection*. Strong link-layer authentication prevents the attackers to send trusted MAC-layer traffic, and is probably the most important component of denial-of-sleep defense. Existing options for implementing link-layer authentication in WSN include TinySec, which is incorporated into current releases of TinyOS, and the authentication algorithms built into IEEE 802.15.4-compliant devices. Anti-replay protection is usually achieved by maintaining a neighbor table of packet sequence numbers. Unfortunately, such a table can become unwieldy even in moderately sized networks. However, network layer neighbor information can be leveraged to limit the number of neighbors that must be tracked to those from which legitimate traffic is expected. In particular, the authors of [32] suggest to use a protocol called CARP that bounds the size of the neighbor table according to the maximum node degree and the number of clusters that are previously configured. Finally, broadcast attack protection allows to detect a denial-of-sleep broadcast attack based on measurements of the ratio of legitimate to malicious traffic, along with the percentage of time that the device is able to sleep.

#### Network and Routing Layer

At the network layer, many attacks can disrupt the network availability. We will describe them one by one, together with specific countermeasures. However, it is worth taking into account that in general security at the network layer highly depends on authentication. Due to resource constraints, authentication in WSNs cannot rely on public key cryptography. Based on symmetric keys and hash functions, Zhang and Subramanian [33] proposed a message authentication approach which adopts a perturbed polynomial-based technique to simultaneously accomplish the goals of lightweight and resilience to a large number of node compromises, immediate authentication, scalability, and non-repudiation.

**Direct Attacks on Routing Information** A direct attack against the routing layer can try to spoof, alter, or replay routing information. By subverting this information the adversary can change to his favor the data flow. An effective countermeasure against the first two problems is to use a message authentication code (MAC). Counters or timestamps can be used to defend against replay attacks [34]. More generally, the authors of [35] proposed two techniques that mitigate the effects of routing misbehavior: the watchdog and the pathrater. The first is used to identify misbehaving nodes, while the second helps routing to avoid these nodes. Similar countermeasures can as well contrast most of the attacks exposed afterwards.

**Hello Flooding** Hello messages are often used to discover neighboring nodes and automatically create a network. Many protocols which use this mechanism make the naive assumption that the sender is within radio range. However, an adversary with a high powered transmitter can corrupt a sensor and make other sensors believe that such a malicious node is in their neighborhood. Data packets routed to the malicious sensor will be indeed sent into oblivion [36], causing both data loss and energy wasting. Fig. 1.1a illustrates such an attack. Generally, a simple countermeasure to the hello flooding attack is to check for bi-directionality of each transmission link. In [37] a method based on signal strength has been proposed to detect and prevent hello flooding attacks.

**Black/Sink Hole Attack** The black hole attack works by inducing the sensors to route all the traffic through a set of compromised nodes, that can then drop (or access) all the routed packets. This attack can be detected by listening to and monitoring transmissions by neighbors, and can be tackled using advanced routing algorithms such as REWARD [38]. Black hole attacks can be even more dangerous when the attacker knows the position of the sink, and tries to become the node used by all other nodes to reach the sink. In this case the attack is called Sink Hole Attack, depicted in Fig. 1.1b. To detect sink holes, the authors of [39] proposed an algorithm that firstly finds a list of suspect nodes, and then identifies the intruder in the list through a network flow graph. However, the sink must flood the network with a request message containing the IDs of the affected nodes, and then these nodes have to answer with specific information regarding the correct path, making the algorithm burdensome. In [40] and [41], two other routing protocols against the sink hole attack have been proposed. However, they are respectively based on the Ad-hoc On-demand Distance Vector (AODV) and the Dynamic Source Routing (DSR) protocols, both not very suitable for WSNs. In [42], an intrusion detection system called MintRoute was proposed. It detects sink hole attacks and can be used with the most widely used routing protocol in sensor network deployments.

**Wormhole Attack** The wormhole attack leverages a fast and powerful connection (often a wired one) between two faraway compromised nodes to subvert routing information. The adversary can tunnel data between the locations of the two nodes, so as to convince other sensors that they know the quickest path to reach the other side of the network. Fig. 1.1c shows this attack. Most existing ad-hoc network routing protocols, without some specific defensive mechanism, will be severely disrupted by this simple attack. A general solution for detecting and countering wormhole attacks has been introduced in [43], based on *packet leashes*. A leash is any information that is added to a packet in order to restrict its validity. Two types of leashes are proposed: geographical and temporal. The former ensure that the recipient of the packet is within a certain distance from the sender, while the latter establishes an upper bound on the packet's lifetime.

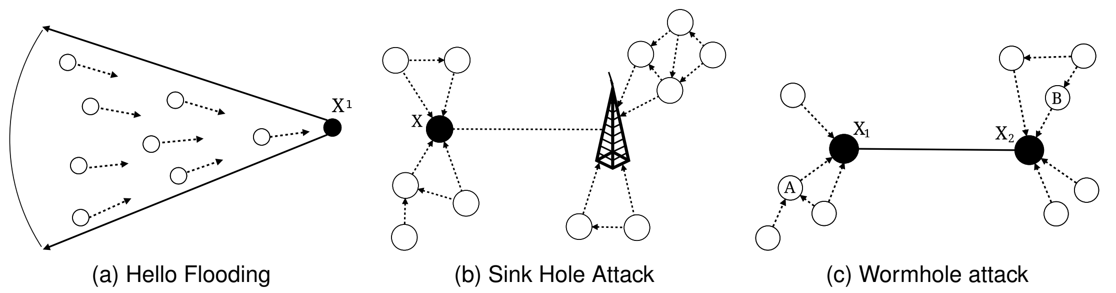


FIGURE 1.1: Some examples of attacks at the network and routing layer

**Selective Forwarding** When a malicious node does not follow the routing protocol, but acts as a filter forwarding certain messages and dropping others, we face a selective forwarding attack [36]. The black hole attack can be seen as a special case of selective forwarding, where all the packets are dropped. In [44] a centralized intrusion detection scheme based on Support Vector Machines (SVMs) and sliding windows is proposed to tackle selective forwarding. In the scheme presented in [45], instead, detection occurs in both the base station and the source nodes, with alarms raised based on multi-hop acknowledgements from intermediate nodes. Finally, selective forwarding can be tackled using redundant schemes like multipath routing [36, 46]: the same packet is sent along multiple paths to increase the probability of reaching its destination.

**Sybil** A Sybil attack consists in a malicious node claiming multiple identities. It was first introduced in peer-to-peer networks [47], but Karlof and Wagner [36] showed it can be a threat in WSNs as well. Fault tolerant schemes, routing and distributed storage algorithms can be easily affected by such an attack. A taxonomy of possible variants of Sybil attacks in WSNs was presented in [48], together with several defensive mechanisms. The main countermeasures are: (i) radio resource testing, that is, asking all nodes to transmit at the same time in a different channel, (ii) key validation for random key pre-distribution, that is, verifying that a node possesses the keys associated to the identity it claims to have, and (iii) position verification, that is, identify Sybil nodes based on the fact that they will appear exactly at the same position.

### Transport Layer

All transport layer protocols can be classified into those that provide congestion control mechanisms, and those that provide reliability [49] of the data transfer. The latter are the most relevant, and their main purpose is to guarantee that every packet loss is detected, and that lost packets are retransmitted until they reach their destination. A reliable transport layer protocol can only detect packet losses if there is some kind of feedback in the system. A scheme can use two types of acknowledgements (*ACK*s): explicit, when a node sends back a confirmation for any packet received, or implicit, when each node verifies the delivery of a packet to a neighbor by overhearing that that neighbor is forwarding the packet. Further, a protocol can use negative acknowledgements (*NACK*s) if nodes are somehow able to realize the non-reception of a packet, and they explicitly send a request for retransmission.

Several transport layer protocols have been explicitly designed for WSNs (*e.g.*, Fusion [50], CODA [51], CCF [52], Siphon [53], ARC [54], Trickle [55], STCP [56], ESRT [57], GARUDA [58], PSFQ [59], DTC [60], RBC [61]).[7] Unfortunately, most of such protocols were designed to ensure reliable communications in the presence of unintentional errors, and not when the network is under attack. Indeed, they fail to provide end-to-end reliability and are subject to increased energy consumption in the presence of an adversary that can replay or forge control packets. When considering attacks at the transport layer, however, we need to assume that the adversary cannot delete both control and data packets (*e.g.*, by jamming), because it would make theoretically impossible to ensure reliable communication [63]. An attack is considered successful if either a packet loss remains undetected or the attacker can permanently prevent the delivery of the packet. Both ACK and NACK-based schemes are vulnerable to injected control packets, but in general ACK-based protocols cannot even ensure reliability, while NACK-based protocols are only vulnerable to energy depleting attacks. Since the latter type of attacks are in practice less relevant, NACK schemes (*i.e.*, PSFQ [59], [58]) may be preferred to ACK schemes (*i.e.*, [60], [61]). NACK schemes are also more suitable for multi-hop communication, but they have two intrinsic weaknesses. On the one hand, the last fragment of a message is theoretically not protected. This problem is in reality easy to solve, by including the total number of fragments in the first transmitted fragment. More importantly, NACK schemes offer no defense against the loss of the whole message, and there is no satisfactory solution at the moment for this problem [64]. Providing authentication at lower layers can solve many of the above cited problems. At least, by authenticating control packets, it would be more difficult for an attacker

[7] A detailed description of all existing protocols is out of the scope of this paper. We remand the interested readers to the corresponding references or to the surveys [49, 62].

to deplete the batteries of the sensors, and thus, to decrease the lifetime of the network. In the following, we will analyze the two main type of attacks to the transport layer [25]: flooding and desynchronization.

**Flooding** Flooding attacks exhaust the memory resources of a sensor, by sending many connection establishment requests to the victim, which consequently allocates resources that maintain state for that connections. To reduce the severity of these attacks, *client puzzles* have been introduced [65]: when a client requires the access to a resource, the server answers with a puzzle that the client has to solve in order to gain the required access. Even if puzzles involve a processing overhead, this is often acceptable with respect to excessive communication. A protocol based on client puzzles and suitable for WSNs has been proposed in [66]. It mitigates DoS attacks against broadcast authentication by leveraging a weak authentication mechanism that uses a key chain.

**Desynchronization** In a desynchronization attack, the adversary forges messages containing bogus sequence numbers or control flags to disrupt an existing connection between two end-points. By continuously causing retransmission requests, this attack can eventually prevent the end-points from exchanging any useful information, other than quickly drain all the power resources of the attacked nodes. The typical and effective countermeasure to this attack is authentication, whether of the header or of the whole packet.

### 1.2.2 Attacks Against Confidentiality and Privacy

The more WSNs become pervasive, the more confidentiality and privacy represent two primary concerns. For example, in military applications confidentiality is a must. On the other hand, in participatory sensing privacy is usually considered a priority with respect to confidentiality. In many other contexts, like automatic health monitoring or commercial applications, privacy and confidentiality are both fundamental [67]. Data confidentiality needs to be enforced through access control policies, to prevent misuse of information by unintended parties. Privacy must be addressed when sensors are not property of the central authority, or in general every time data gathering may involve contextual information which monitored entities do not want to share with the network authority. Confidentiality and privacy issues involve even ethical or legal aspects. However, we will only discuss the technological solutions to enforce such security requirements in WSNs.

**Eavesdropping** If end-to-end communications are not protected, anyone is able to discover the communication content by simply eavesdropping on the network's radio frequency range.

This way, even passive outsider adversaries can steal private or sensitive information. The standard approach to face this basic attack is cryptography: data are encrypted so that only intended recipients can decrypt the message. Some non-cryptographic approaches have been discussed in the literature [68], but their interest is limited when the adversary can only eavesdrop. Because of sensors having limited computational power, symmetric-key encryption is preferable to public-key. The most used and cited scheme based on symmetric keys is SPINS, introduced in [34]. SPINS is a suite of security protocols optimized for WSNs and consisting of two building blocks: the first assures data confidentiality, two-party data authentication and data freshness, while the second provides an efficient broadcast authentication mechanism. Usual key management schemes for symmetric protocols are unfeasible in many WSNs applications, so most encryption schemes rely on key pre-distribution. When two sensors want to communicate securely, they must first execute a key-discovery phase, to find out which keys they share, and then compute a session key based on such shared keys [69, 70]. However, the authors of [71] highlight a major problem of keys pre-distribution: an attacker can easily obtain a large number of keys by capturing a small fraction of nodes, and leverage such keys to disrupt the authentication mechanism. In particular, if the sink is mobile and cannot therefore be identified by its position, the adversary can deploy a replicated sink preloaded with the compromised keys, which many sensors will confuse with the legitimate sink. To address this issue, the authors propose a new framework relying on two separate key pools, one for the mobile sink to access the network, and one for pairwise key establishment between the sensors. Finally, in WSNs data collected by sensors are usually processed and aggregated at each intermediate node before they reach the sink, to achieve power efficiency by reducing data redundancy and minimizing bandwidth usage. *Data aggregation* is unfortunately in conflict with data confidentiality. The former requires encryption between the originating node and the sink, but apparently intermediate nodes need to access the cleartext, in order to aggregate data. Homomorphic encryption is the natural solution to overcome this impasse, as proposed in [72] and [73].[8] The scheme presented in [72] allow to aggregate data in a confidential and efficient way, relying on a simple but provably secure homomorphic encryption function. The scheme in [73] is able to provide both confidentiality and integrity of the aggregated data.

**Traffic Analysis** Encryption alone is not enough to assure *secrecy in a broader sense*. An adversary can analyze overheard data traffic to gain important information about the network topology and the sensed events. Just leveraging traffic analysis, the adversary can identify sensors with special roles [75], or run targeted attacks designed to maximize harm. Deng *et al.* proposed countermeasures against traffic analysis attacks that seek to locate the base station [75]. Recently, Wadaa *et al.* proposed schemes to randomize communications during the network set-up phase, to protect anonymity of sensor network infrastructure [76].

[8]We suggest [74] for a review of aggregation techniques enforcing confidentiality in WSNs.

### 1.2.3 Attacks Against Data Integrity

Data integrity is violated when the adversary corrupts records, and the sink is not able to restore the original sensed data, or at least to detect that data have been manipulated. The simplest attack to compromise sensed data is data erasure, that is, delete any trace of a specific information before it reaches the sink. However, in our classification of security requirements, we considered *data survivability* (the common way to denote resilience to data erasure) a segment of service integrity, rather than data integrity. In standard WSNs, data are off-loaded to the sink as soon as possible, so data erasure requires either compromising the originating sensor before it sends the target information, or intercepting such information along the routing path towards the sink. The former strategy can be easily tackled by letting routing start as soon as data are gathered. The latter is usually implemented using black/sink hole attacks, which can be countered as described in Section 1.2.1. To face more subtle threats to data integrity, tamper-resistance and authentication represent the two basic approaches. On the one hand, if the adversary gains full control of a sensor, fake data injected by that sensor will look legitimate to the sink. On the other hand, if authentication is not used at all, any outsider adversary can alter messages exchanged in the network by simply implementing a man-in-the-middle attack. In general, the success rate of an attack to data integrity depends on the ability of the adversary to capitalize on its resources to circumvent authentication mechanisms. Along this line, *node replication* is the main approach to maximize the impact of sensors corruption. Other active attacks are instead based on *packet injection, replication, and alteration*.

**Node Replication** When an adversary captures a sensor without being detected, he can use that sensor to inject authenticated, but fake, data. Even if sensors used in typical WSNs are not tamper-proof (mainly for cost reasons), in most application settings the number of sensors that an adversary can concurrently control is however limited. To boost the attack, the adversary can clone the corrupted sensors and insert the replicas in the network. Even if the adversary compromises a single node, he can generate enough replicas to subvert voting or data aggregation protocols. To contrast replication, the network should realize that two different nodes are claiming the same identity. Unfortunately, the distributed nature of most WSNs makes detection challenging when clones of the same sensor are deployed faraway from each other. Centralized monitoring [70, 77] can be a solution: all nodes in the network periodically transfer to a central entity a list of their neighbors, including nodes ID and location. If the same node claims two (or more) conflicting locations, the sensor is considered corrupted and all its credentials are revoked. However, centralized approaches have two drawbacks: the introduction of a single point of failure, and the communication overhead incurred by the nodes that surround the central entity. Emergent properties have been used in contrast to centralized monitoring in [78].

The authors proposed two algorithms extremely resilient to active attacks, and both trying to minimize power consumption by limiting communication.

**Packet Injection, Replication and Alteration** To modify data gathered by the network, the adversary has three main alternatives: inject completely false data, replicate previously captured packets, or intercept messages and alter their content. All these attacks can be easily run by insiders, but if the adversary is an outsider they require to break the authentication mechanisms to varying degrees. Injection requires forging from scratch a message that must be indistinguishable from legitimate ones. Replication uses already authenticated massages, but counters or timestamps used to avoid replay attacks need to be counterfeited. Alteration is in general as difficult as injection, but it can result sensibly easier when homomorphic encryption/authentication is used (*e.g.*, for data aggregation). Generally, standard asymmetric authentication protocols are not suitable for WSNs, and are replaced by schemes relying on symmetric keys [70, 77, 79, 80]. In particular, $\mu$TESLA [34] is the "micro" version of the Timed Efficient Stream Loss-tolerant Authentication (TESLA) scheme proposed in [81]. It is based on TESLA, but key update and initial authentication are modified to fit for WSNs. The main idea of $\mu$TESLA is to use a one-way hash function $F$ to form an "inverse" key chain: the sender chooses the last key $K_n$ of the chain randomly, and applies $F$ repeatedly to compute "previous" keys as $K_i = F(K_{i+1})$, for $i = n - 1$ down to 1. A key is published some time after the corresponding message is sent (therefore, $\mu$TESLA requires loosely synchronization between the sensors and the base station, and that each node knows an upper bound on the maximum synchronization error). Since previous keys can be verified through the current key, while the current key cannot be computed from previous keys, an attacker cannot forge keys and authenticate messages. The authors use MD5 as the one-way hash function in TESLA and $\mu$TESLA. However, when the adversary corrupts a number of sensors, he can inject fake data which will be correctly authenticated, regardless of the robustness of the cryptographic schemes used. In [82], the authors propose BECAN, a bandwidth-efficient cooperative authentication scheme for filtering injected false data. Such scheme is able to detect the majority of fake data during the routing path to the sink with minor extra overheads at the en-route nodes, obtaining a remarkable reduction of the burden of the sink. Finally, things become even harder when data are aggregated during the routing path. In this case, the authors of [83] propose a secure extension of the robust (against unintentional failures) but insecure (against fake data injection) *synopsis diffusion* algorithm presented in [84]. In particular, the extension is based on a novel lightweight verification algorithm by which the base station can determine if the computed aggregate includes any false contribution.

## 1.3 Summary

In this chapter, we introduced the WSN paradigm, describing its main features, vulnerabilities and security requirements. Then, we discussed the main security threats and countermeasures in WSNs, classifying attacks according to their *target*. Depending on the service provided, secure WSNs need defensive mechanisms to protect: (i) network availability and service integrity, (ii) data confidentiality and privacy, and/or (iii) data integrity. When dealing with network and service reliability, it is particularly useful to distinguish the threats based on the attacked *layer*, which sensibly affects the nature of the attack (and of the corresponding responses). Security mechanisms must perform at each layer, from the physical, to the link, the network, and the transport layer. The profound attention paid to WSN is due to its relevance in the scenario of automated environmental sensing and monitoring: most of the issues discussed for WSNs extend to DSNs and other more specific models, which however (as we will see in the next chapters) introduce additional and specific security requirements.

# 2 Security in Distributed Sensor Networks

Designing systems able to automatically interact with the environment is one of the main challenges currently addressed by the research community. When dealing with sensing systems, the principal paradigm in the literature is the Wireless Sensor Network (WSN), comprehensively discussed in Chapter 1. Thanks to their ability to offer low cost solutions to a huge variety of real problems, WSNs are frequently used for a wide range of civilian, industrial and military applications [7]. However, the WSN model is not flexible enough to capture characteristics proper to many application settings. In particular, there are two main features that demand considerable attention: the degree of attendance of the sink, and the level of mobility of the nodes. Unattended Wireless Sensor Networks (UWSNs) were specifically introduced to describe all contexts where the assumption of an always present sink would be unrealistic. Similarly, Mobile Wireless Sensor Networks (MWSNs) model those WSNs whose sensors are free to roam in the monitored environment. When the network is, at the same time, both mobile and unattended, it is usually referred to ad Mobile Unattended Wireless Sensor Network (MUWSN).

To reduce deployment and managing costs, it is highly desirable to devise monitoring solutions that do not require the deployment of specific networks. Along this line, relying on emerging technologies, new paradigms were recently introduced to model collaborative information sensing and elaboration in urban scenarios. The rationale is to leverage the mass diffusion of sensor-equipped devices (*e.g.*, smartphones and tablets) to subsidize users collaboration to the process of data gathering, so as to make now feasible and low-cost tasks considered futuristic only a few years ago [85]. In this context, the referential model is Participatory Sensing[1] (PS) [86–88], which, generally speaking, refers to the ensemble of technologies and algorithms enabling a community to contribute to the process of sensing and collecting information from the environment, to form a (shared) body of knowledge.

A Distributed Sensor Network (DSN) is a general model capturing the communal issues and requirements of mobile unattended sensing systems, such as MUWSN and PS. DSNs are in fact defined as *ad-hoc mobile unattended networks that include sensor nodes with limited computation and communication capabilities* [89]. In this chapter, we present a detailed description of DSNs, underlining the main reasons that make security in DSNs a stimulating and challenging task. In particular, in Section 2.1 we focus on UWSNs, to provide a comprehensive overview of the main issues related to the lack of a central control. In UWSNs, sensed information cannot be promptly offloaded to any trusted entity, but must be securely stored in the network in a distributed way. This model offers potentialities that centralized networks fail to provide, but poses stimulating security challenges. Then, in Section 2.2, we describe how mobility and users collaboration in MUWSNs and PS introduce further problematics, but also represent potential resources for the development of novel security schemes. We explain why the more general model of DSNs captures characteristics communal to both paradigms, and we summarize the requirements of such networks, to pave the way for the solution proposed in Chapter 3.

## 2.1 Unattended Wireless Sensor Networks

Unattended Wireless Sensor Networks (UWSNs) were introduced in 2007 [90], to capture all settings where the assumption of a constant data sink would be unrealistic. A simple analysis of contemporary applications of WSNs immediately highlights the importance of this more specific model. In many cases, the inaccessibility of the monitored area, and the technical problems that arise to connect the sink with the sensors, are themselves sufficient reasons not to allow the use of a constantly available sink. Perfect examples in this sense are all underground or submarine networks, involved in the monitoring of critical infrastructures (*e.g.*, in oil pipelines, a network can be required to monitor that the pumps are working correctly and that leakages of oil are not present [91]). In most military applications, from the exploration of a battlefield, to the surveillance of a harbor, the deployment scenario is not only hardly accessible, but even extremely dangerous. Finally, in some cases, like wildlife monitoring (where the network is appointed to control animals health status and detect poaching), the size of the protected area, and the difficulty of hiding a sink, can motivate the requirement for an itinerant sink [92]. In all

[1]PS is sometimes referred to as *people-centric*, *urban*, or *opportunistic* sensing.

these cases, an intermittent sink is often the only viable alternative, and an UWSN can be used to keep tabs on an infrastructure that cannot be monitored by traditional systems.

The absence of a constant, trusted, central authority, able to both monitor the network and gather sensed data in (quasi) real time, makes data security in UWSNs more challenging than in traditional WSNs. In Chapter 1, we saw that data integrity and confidentiality in WSNs depend primarily on intrusion detection, encryption, authentication, and multipath routing. In fact, in WSNs the sink can supervise the network and (almost) continuously check for sensors malfunctioning or capture. Sensed data are promptly sent to the sink, and do not need to be securely stored in the network. To the contrary, in UWSNs it is natural to assume that the adversary can leverage the absence of the sink to compromise sensors, read, delete or alter sensed and stored information, and disappear without leaving any evidence of its illegal behavior (*e.g.*, in the case of pipeline monitoring, an adversary can steal oil from the pipeline, and corrupt data collected by the nearest monitoring sensors). In other words, intrusion resistance is unfeasible, and the attention is moved to *intrusion detection and recovery*. Real-time delivery of the sensed data may not be critical, but it is fundamental that the sink can indeed collect the sensed data, and that it receives such data as soon as possible. Even when the sink cannot gain direct access to the sensor that actually detected an event, the sensors must therefore collaborate to not only avoid data loss, but also to facilitate data collection. Data sensed while the sink is away are particularly exposed, and it is necessary to enforce *data survivability, confidentiality and authentication* using *secure distributed data processing and storage schemes*. If a security mechanism is not provided, the sink, and therefore the monitoring system, will not ever know that a piece of information has been deleted or compromised by an adversary. However, because of the limited resources of sensor nodes, this must be done without incurring in excessive energy consumption.

### 2.1.1 Security Threats to UWSNs

Before discussing more in detail security threats and countermeasures for UWSNs, let us better discuss the *adversary model*, the *cryptographic techniques* that can be used, and the *security requirements* to ensure.

**Adversary Model** As we already pointed out, in UWSNs it is natural to assume the presence of an *active* outsider attacker, able to compromise nodes during the absence of the sink without leaving traces. However, the number of sensors that the adversary can corrupt in each interval is limited, since otherwise it could gain complete control of the network and irreparably threaten security. For a similar active but limited adversary, it is fundamental to distinguish between *mobile* or *stationary* attacks. Independently from the fact that the network itself is mobile or stationary, the distinction between mobile and stationary adversaries aims at capturing the ability of the attacker to compromise different sets of sensors. Depending on the adopted model, a mobile adversary can physically move and compromise sensors deployed around him, or somehow "jump" from a set of sensors to another one. A stationary attacker instead chooses a subset of sensors at the very beginning of his attack without changing his target thereafter.

**Cryptographic Techniques** As in more general WSNs, symmetric encryption is usually used for data confidentiality and authentication purposes. Simple cryptographic functions are preferable, like one-way hash functions [93] and efficient symmetric schemes such as AES [94] or Skipjack [95]. Skipjack is in particular used for WSNs in the TinySec scheme [96] due to its power efficiency. However, the more stringent security requirements sometimes push towards public key cryptography, which gives more guarantees at the cost of a major resource consumption.

**Security Requirements** The three main security requirements in UWSNs are: *Data Survivability and Confidentiality*, *Intrusion Detection and Recovery*, and *Data Authentication*. Since data cannot be off-loaded to the sink in real time, they reside in the network for a longer time than in typical WSNs. This exposes data, raising concern for their integrity and confidentiality. In particular, data *survivability* becomes a major issue because the main objective of an adversary is often to delete sensed data before they reach the sink. Intervals between successive sink visits represent periods of vulnerability, and therefore they give a boost to the activities of an adversary. Frequent intrusions become a necessary assumption, and it is fundamental to be able to detect when nodes are not working as intended, or (even better) to recover compromised sensors. In particular, *self healing schemes* are a remarkable mechanism to restore secure communication with previously corrupted nodes. Finally, the attention paid to data authentication in UWSNs is mainly due to a simple observation: UWSNs cannot use standard data authentication mechanisms that rely on a centralized entity, otherwise, with sufficient time between sink visits, an adversary could easily compromise sensors collected data. In the sequel, we will categorize the main threats based on the security requirements they affect, and describe the corresponding solutions proposed in the literature.

#### 2.1.1.1 Data Survivability and Confidentiality

In UWSNs, sensors inability to directly off-load data to the sink makes it easy for an adversary to perform focused attacks aimed at deleting certain target data. Further, the fact itself that UWSNs are often deployed in hostile environments means that it is extremely reasonable that the network is performing some sort of surveillance duties. Consequently, *data survivability* is usually considered the main concern. In this scenario, it is normal to assume a mobile adversary who is actively hunting a certain data item, and who is not afraid to delete/erase any other data he/she finds.

In [90], the authors proposed a better characterization of the adversary as:

- *Lazy*, when the attacker is stationary, and at the beginning of the protocol chooses $k$ nodes to compromise, without ever changing his target thereafter;
- *Frantic*, when the attacker is mobile, and captures a different subset of $k$ randomly chosen nodes each time the sink leaves the network;
- *Smart*, when the attacker is mobile, but only skips between two pre-selected sets of nodes, each of size $k$.

In the paper, three simple non-cryptographic survival strategies were studied:

- *DO-NOTHING* is the trivial survival strategy, where each sensor simply stores its own sensed data, waiting for the sink arrival;
- *MOVE-ONCE* prescribes that data are moved just once to a new sensor randomly picked among the whole network;
- *KEEP-MOVING* requires that data are continuously and randomly moved from sensor to sensor.

The analysis of all possible attack-survival strategy combinations conducted in [90] highlights that: (i) the DO-NOTHING survival strategy is useless, (ii) when MOVE-ONCE is implemented, a FRANTIC adversary is the most advantaged, and (iii) when KEEP-MOVING is used, a SMART attacker is the most effective one. In [97], resilience to an adversary dubbed ERASER, who wants to indiscriminately erase any information, is analyzed. Surprisingly, the best survival strategy results the DO-NOTHING: moving data only helps the ERASER to encounter and erase all data faster. However, the authors investigated the effects of data replication, showing that with replication the KEEP-MOVING strategy becomes the best solution against an ERASER. In [98], encryption is used to hide contextual information (*e.g.*, the origin and time of collection of a packet), other than the content of sensed data. The rationale is to prevent the adversary from recognizing target data, forcing him to erase data blindly (like the ERASER attacker). An interesting additional result of this analysis is that public key cryptography allows to obtain the same level of security of continuously moving data, by combining moving data just once and re-encrypting them. Replication is deeply discussed in [4], where a pure controlled epidemic technique is used to provide a trade-off between data survivability, optimal usage of sensor resources, and a fast and predictable collecting time. The authors prove that by estimating the maximal power of an attacker it is possible to set up a probabilistic bound on the survivability of the data. Despite the very interesting approach, the results exhibited in [4] were however partially incomplete. We carried on a more comprehensive analysis of the application of epidemic models to the survivability of information in UWSNs. As described in the Introduction, the work, where we show how to set system parameters so as to perfectly tune data replication rate, will be submitted soon.

Apart from the necessity of limiting energy consumption, replication poses a more serious challenge: while enforcing survivability, replicating data harms data confidentiality. In fact, the more replicas of a data are generated, the more easy is for an active adversary to find (and compromise) a sensor which is storing one of such replicas. Alternative non-cryptographic solutions for secure and distributed storage in UWSNs were investigated in [99]. The authors proposed two algorithms: DS-PADV, to protect against adversaries which do not know where the target information is stored, and DS-RADV, a more secure but burdensome scheme to defend from reactive adversary which choose nodes after identifying the target. However, the most promising solution to ensure both survivability and confidentiality of sensed data in UWSNs is represented by secret sharing based schemes. In [100], the authors showed how a similar solution can maximize communication and storage efficiency and data survival degree. They also introduced an enhanced scheme based on network coding to further improve the power consumption efficiency of communication. However, the aforementioned papers did not fully clarify the importance of secret sharing schemes for distributed secure storage in UWSNs. For this reason, in [3] and [2] we proposed a detailed analysis of secret sharing schemes in Distributed Sensor Networks (DSNs), which will be presented in detail in Chapter 3. In [3], probabilistic bounds are introduced to predict the amount of sensed data that can be reconstructed, only based on the shares stored by a given portion of the network. Such bounds show that secret sharing can indeed provide the desired trade-off between survivability and confidentiality in UWSNs.

#### 2.1.1.2 Intrusion Detection and Recovery

In our definition of the adversary model, we stated that it is necessary to assume that the capabilities of the adversary are limited, in that it can only capture a small number of sensors during each period of absence of the sink. However, if the adversary can keep control of sensors captured previously, he will eventually gain control of the whole network in any case. For this reason, it is fundamental to detect intrusions, and to try to recover as many corrupted sensors as possible. Data stored in a corrupted sensor are irremediably lost. However, we can restore a secure keyring to prevent the adversary to access data sensed or received by that sensor in the future/past, or to forge new authenticated data. In other words, we are interested in *backward* and *forward secrecy* of the keys. Forward secrecy can be easily obtained through periodic key evolution [101]. In contrast, backward secrecy is much harder to attain since it relies on a source of randomness that the adversary must not control. Solutions based on asymmetric key pre-distribution have been proposed [102], but their feasibility is limited due to the computational cost of asymmetric cryptography. In [103], the authors introduced scheme called DISH, based on symmetric keys. It leverages sensor collaboration to recover from compromise, and maintains the secrecy of collected data. It provides both backward and forward secrecy using a "sponsor" technique: healthy nodes sponsor sick nodes to make them healthy again. Sponsorship in this context means to supply a pseudo-random value to the sponsored node, which the latter uses to renew cryptographic keys. More precisely, in each round, each node requires values from $t$ sponsors, and it uses these values in the next round to update its own symmetric key. The authors consider a mobile adversary that can compromise up to $k$ nodes in each time interval. Two possible strategies are analyzed: the *Trivial Adversary* and the *Smart Adversary*. The former tries to compromise in each time interval a new set of randomly selected sensors that are not yet compromised. The latter selects the subset of nodes to be compromised in such a way to disrupt the sponsor mechanism, preferring to compromise the sponsors of a sick node in order to maintain it sick. DISH successfully mitigates the effect of sensor compromise. However, it requires many messages to be exchanged in each round. To overcome this issue POSH was presented in [104]. The idea is similar to DISH, but it differs in one main feature: sponsors push instead of being pulled. In other words, instead of nodes explicitly requiring the contribution of $t$ sponsor nodes, the latter voluntarily send their contributions. In this way, the request messages are no longer used, hence decreasing the overall energy consumption.

Previously cited schemes consider an attacker that can compromise up to a fixed number of sensors in each round, randomly picked in the whole network. A more realistic hypothesis is an adversary that can compromise only sensors within its communication or action range. This adversary is analyzed in [105], where the attacker can control a fixed portion of the network deployment area, and compromise all sensors that move within it following a particular mobility model, such as the random way point, or the random jump. The proposed scheme is based on public key cryptography, but it uses an evolution mechanism based on node collaboration to generate one-time symmetric random keys. In particular, the scheme leverages the mobility of the nodes in a way similar to the push mechanism used in POSH. In each round, nodes broadcast a "contribution" that is then used by their neighbors to calculate the next one-time symmetric random key. Another scheme that uses sensor mobility is the one proposed in [106]. However, in this work a different adversary is analyzed, able to roam the network and choose in each round a new portion of the deployment area to compromise. The proposed protocol is similar to the one presented in [105], but the mobility of the adversary leads to different results. The authors show that the proposed scheme depends on: (i) the portion of the deployment surface controlled by the adversary, (ii) sensors mobility model, and (iii) the density of the network. Analyses and simulations show that the best self-healing performances are achieved when adopting a sensor mobility model that provides high variability in sensors neighborhoods.

#### 2.1.1.3 Authentication

Authentication for unattended sensors was first investigated in [107], where the authors introduced a technique for data aggregation providing forward-secure authentication. However, the scenario analyzed in [107] in not really a network, since communications among sensors are not considered at all. The first scheme that explicitly provides authentication in UWSNs was proposed in [108]. The authors consider a mobile adversary that attempts to replace authentic data with data of his choice. They introduce two techniques that leverage sensor cooperation, and that rely on symmetric cryptography: Co-MAC and ExCo. In Co-MAC, which stands for "Cooperative MAC", each information is authenticated either by the node that sensed it, and by a set of co-authenticators. The co-authenticators are selected using a Pseudo Random Number Generator (PRNG), and are required to keep the MACs of all data they authenticated. The PRNG relies on a secret seed shared with the sink, which consequently knows which sensors store the MACs corresponding to any data sensed at any round. ExCo stands for "Extensive Cooperation", and uses a different approach: sensors do not send their data, but they send the MAC of their data to the co-authenticators. When sensors serve as co-authenticator for multiple MACs, it can bundle all such MACs into a single authentication tag. In both Co-MAC and ExCo, to alter authenticated data, the mobile adversary needs to compromise both the originating sensor and all the co-authenticators. The authors show that the probability of a successful attack rapidly drop as the number of the co-authenticators grows. ExCo was finally extended in [109], introducing a mechanism to dynamically adapt the number of co-authenticators.

## 2.2 Mobility in Sensing Systems: Introducing Distributed Sensor Networks

The first model proposed to describe monitoring activities performed by mobile sensors is the Mobile Wireless Sensor Network (MWSN). In [110], the authors surveyed the security challenges posed by MWSNs, underlining a lack of specific literature, and that mobility is often treated as an obstruction to the design and analysis of security schemes. Yet, mobile networks are fairly common nowadays and, with properly designed protocols, mobility can help increasing connectivity [111], improving coverage and energy saving in WSNs [112], or enhancing security in general in Ad-Hoc networks [113]. The mobility of the nodes may depend on a precise design choice, but is often imposed by the intrinsic characteristics of the network. This happens particularly in unattended contexts, such as wildlife monitoring, where sensors are attached to alive animals, or underwater networks, whose nodes are subject to wave motion and currents.

One of the main models characterized by both unpredictable movements of the sensors, and absence of any centralized control is Participatory Sensing (PS). The implementation of PS is in fact conditioned by a main limitation: sensors are not property of a single entity. This means that nodes behavior and position cannot be fully predicted, sensors are not trusted, coordination is harder, and users privacy must be guaranteed to subsidize their collaboration. The state of the art in PS security is well surveyed in several papers, which summarize possible issues [114], applications and solutions [85]. As reported in [115], security challenges in PS are usually considered different from those posed by MUWSNs. The attention of most researchers [116–120] is mainly paid to schemes enforcing privacy among the parties of the sensing infrastructure. In particular, the central authority should not acquire any sensitive information about the data collectors and the queriers. The main topics of our analysis, data confidentiality and availability, are often reputed of minor importance in PS, because of the general superior energy and communication technologies available to smartphones and similar devices with respect to traditional sensor nodes. Some papers do recognize that, PS networks being unattended, nodes can be attacked, sensed data can be revealed, and malicious nodes can inject fake data into the system. Along this line, in [116] the authors propose negative surveys[2] to reliably reconstruct the probability density function of the original sensed data, in a privacy-preserving way resilient to data poisoning. With similar goals – ensure integrity of the aggregated data, while still providing users privacy – the authors of [117] suggest homomorphic message authentication codes and data slicing and mixing among nodes. However, standard security protocols are usually considered feasible in PS, and real-time record of the sensed data is only traded-off against privacy requirements (direct communication of the sensed data to the central authority may expose information about the users). Nevertheless, especially when sensors are asked to collect data *continuously*, it is naive to enforce data security based on sensors real-time response. Long-range communication technologies require both energy consumption and bandwidth utilization, and data elaboration and transmission would affect the performances of the sensing devices in an intrusive way for the users. Treating security in PS similarly to the more general context of DSNs allows to optimize both efficiency and security, helping PS to become *transparent* to the users. Further, a similar approach opens the way to the involvement of a wider range of devices to the process of data gathering.

Distributed Sensor Networks (DSNs) are composed by mobile untrusted sensors, and characterized by the absence of any centralized control. Similarly to UWSNs, one or more trusted entities, called *sinks*, sporadically access the network to retrieve as much sensed information as possible. DSNs are vulnerable to several security threats, particularly because of their distributed nature and of sensor nodes having limited usable resources. The main issues, together with

[2]In a negative survey, sensors report a random value picked in the complement set of the actually sensed value.

possible counteractions, have been widely studied and are surveyed in a number of works [121–123]. The resources available to sensors can sensibly vary according to the specific application setting. However, information security in DSNs cannot rely on standard techniques, such as tamper-proof hardware (which cannot be installed in low-cost or non proprietary sensors), standard secure routing protocols (that would extinguish sensors' batteries too fast, due to the considerable energy required by data transmission and reception), or public-key cryptography (which involves excessive computational costs). Thus far, the research community separately addressed secure key-management algorithms needed to enforce symmetric-key cryptography, distributed intrusion and capture detection schemes able to limit active attacks, or carefully designed energy-efficient routing protocols [121]. Unfortunately, the solutions proposed so far do not seem to be rooted on some general principles that could provide all security requirements at once. DSN requires a new and deep understanding of information handling, going beyond traditional security mechanisms which require large computation and communication overhead. Our purpose is exactly to show how a high-level, distributed, light-weight information handling scheme can ensure data confidentiality and availability, as well as enhance the level of protection against fake data injection and information traceability. Designing a similar resource-saving secure data storage scheme is crucial to enable very heterogeneous types of networks to contribute to environmental monitoring activities, without sacrificing security.

Finally, let us give a brief overview of how location privacy in DSNs has been addressed so far in the literature. The authors of [124] present a useful survey on privacy preservation in DSNs, where location privacy is divided into two main categories according to what we want to protect: data source or data sink. In DSNs, the sink can follow two different strategies: randomly explore the network at every round, or periodically cover the same path. In mobile contexts, not even the latter approach exposes the sink, because information about the past behavior of the sink can hardly be deduced by traffic analysis. The attention is therefore totally directed to source-location privacy, enforced based on two major techniques. The first strategy consists in confusing the information available to the adversary by routing fake packets mixed with the real ones. This is mainly implemented through *dummy injection* [125, 126] or *fake data sources* [127]. The second strategy relies on making a routing path difficult to track back. It is primarily realized through *flooding schemes*, where each sensor broadcasts the data it senses or receives to all of its neighbors, deterministically [128] or probabilistically [125]. All the former techniques could in principle be applied in DSNs, but they require a remarkable amount of unnecessary messages, with consequent power consumption, though having a limited efficacy. Our solution is based on the same concepts, confusing the source and reducing the correlation between the current and original position of data, but it is realized in a much more efficient way, which only relies on local communication and nodes mobility.

## 2.3 Summary

In this chapter, we introduced information security in DSNs. Since the main features of DSNs are the absence of a (constantly available) central authority, and the (random) mobility of the sensor nodes, we described the main models of unattended and/or mobile sensing systems considered in the literature. We started with a characterization of UWSNs, to underline all threats and security requirements associated to unattended networks, and to summarize the solutions proposed so far. We delineated more precisely how the intermittent presence of the sink affects the typical capabilities of the attacker, and the range of feasible cryptographic techniques. Then we focused on MWSNs and PS, which represent two fundamental paradigms for mobile and distributed sensing applications, where nevertheless the potential benefits of mobility were rarely directly addressed. The study of UWSNs, MWSNs and PS allowed us to identify the main goals and limitations that must be taken in mind when designing a secure data handling scheme for DSNs. In particular, we observed that security cannot really rely on cryptography when the network is constantly exposed and the topology of the network is continuously changing, but that important results can be obtained through distributed collaboration and storage. This analysis paves the way for our solution, presented in the next chapter.

# 3 Information Sharing and Diffusion in Distributed Sensor Networks

*Secret sharing schemes* [129] represent probably the most interesting solution proposed so far for DSNs security. They respond to the idea that sensed data must be available after a number of nodes have offloaded their data, and not as soon as one of them gathered them. This means that data can be elaborated, exchanged and diffused, with the aim to maximize the amount of information that the provider can deduce *after re-elaborating everything it collected.* Secret sharing was already introduced for MANETs in [130], and firstly applied to UWSNs in [131], providing resilience to data invalidation. In [132] and [133], the authors defined two new schemes for distributed data storage and retrieval in UWSNs, based on the combination of secret sharing and encryption. However, all the described approaches lack of a comprehensive analysis of how other characteristics of the network, mobility over all, can be leveraged to improve the efficiency and effectiveness of the scheme. More importantly, all those papers restrict themselves to a discussion of the idea and the expected results, sometimes supported by simulations and experiments, but never by a deep theoretical analysis of the effectiveness of the scheme, which instead is what we do in this chapter.

In the following, we describe a novel solution that takes advantage of mobility to efficiently implement a secure and distributed data storage scheme. Based on a local implementation of secret sharing and on information diffusion, our protocol addresses, at once, all the security needs of DSNs. In particular, to handle the delicate but necessary trade-off between data availability and confidentiality, we exhibit precise bounds binding the accessed fraction of the network to the amount of information recovered by both the sink and the adversary. The performances of the network under three different mobility models are compared both analytically and through extended simulations.

The purpose of this chapter is to introduce and enforce the idea that a secure information handling scheme should consider the information sensed by the whole network as a unique entity, instead of focusing on the data sensed by each node individually. Along this line, we propose a solution based on the concepts of *information sharing and diffusion*. To limit energy and bandwidth consumption, we rely on *distributed collaboration* (data are elaborated and shared only *locally*), and cleverly leverage nodes mobility. More precisely, we suggest that, right after information detection, each node locally implements a $(k,n)$ threshold secret sharing scheme [129] on the sensed data, deriving $n$ shares such that any $k$ of them are necessary and sufficient to reconstruct such data. All sensors instantly send the $n$ generated shares to as many randomly selected neighbors, so that each sensor receives and stores approximately $n$ shares of different sensed data. Since sensors continue to roam freely in the monitored area, as time goes by their position says less and less about what pieces of information they carry. *Quantitatively*, any node carries more information than is contained in any of the sensed data (to reconstruct which only $k$ shares are necessary), though it actually carries *no* information (but only *pieces* of information, that are meaningless by themselves). On the whole, the *quantity* and the *content* of the information that can be recovered from a given fraction of the network now only depends on the *size* of that fraction, and not on which nodes are concretely involved. A similar approach is expected to enforce availability without sacrificing confidentiality, relying on the righteous assumption that the sinks can access a sensibly larger portion of the network than any realistic adversary. Similarly, information sharing enhances resistance against fake data injection, provided that the workload of the sinks can be tuned on the estimated counterfeiting ability of the adversary. Finally, the more information is shared and the faster the nodes move, the harder it becomes for the adversary to infer on data source location.

## 3.1 System Model

Before we discuss our solution in detail, we need to better characterize our system model. We will separately describe our network model (Section 3.1.1), our adversary model (Section 3.1.2), and our goals (Section 3.1.3).

### 3.1.1 The Network

We assume that $N$ sensor nodes, denoted $s_1, \ldots, s_N$, are randomly deployed in an $L \times L$ square area $A$. Each node $s_i$ moves randomly in $A$, monitoring the environment in a radius $r$ around it. The data sensed by $s_i$ at time $t$ are denoted as $D(i,t) \in \mathbb{F}_q$, for a suitable $q$. In general, the number of sinks and their behavior (passive/active, static/mobile, periodic/random path, etc.) could sensibly vary. To simplify the analysis, we assume that sensed data are collected by a single mobile trusted sink, which sporadically explores a random portion of $A$, and recovers all the data stored by the nodes in its communication range.

The only feature of the network that poses serious modeling issues is sensors mobility. It is natural to assume that sensors move randomly and independently from each other, but it is very challenging to describe their movements in a way that offers a reasonable trade-off between realism and ease of treatment. Our choice came down on comparing three different mobility models, confident to provide an acceptable and synthetic overview of the possible application settings. Such models, much used in the literature [110–113], are the following:

**IID** In the IID mobility model, time is discretized into rounds, and the positions occupied by a sensor over time are independent identically distributed (i.i.d.) random variables. In other words, at regular intervals the location of each sensor is updated by a draw from the same probability distribution $\mathcal{P}$ over $A$. We will take $\mathcal{P}$ as the uniform distribution over $A$, as it is common practice. The IID is the simplest possible model, and serves as a benchmark to which the other models can be compared.

**Random Walk** The simple two dimensional RW is a discrete-time mobility model. If $(x_i^t, y_i^t)$ are the coordinates of node $s_i$ at time $t$, its position $(x_i^{t+1}, y_i^{t+1})$ at time $t+1$ is uniformly picked among the four points $(x_i^t \pm l, y_i^t)$ or $(x_i^t, y_i^t \pm l)$. $l$ is a parameter of the scheme, used to define the speed of the sensors.

**Random WayPoint** The RWP is the most realistic mobility model we consider. Time is continuous, and sensors motion is simulated by a sequence of independent rectilinear movements. The RWP uses two parameters $v_{\min}$ and $v_{\max}$ to describe the minimum and maximum speed to which nodes can move. For each movement, a destination point $Q$ and a speed $v$ are uniformly picked in $A$ and in the interval $(v_{\min}, v_{\max})$, respectively. The node moves at speed $v$ until it gets to $Q$, then new destination and speed are randomly selected, and so forth.

### 3.1.2 The Adversary

Our network model imposes to assume the presence of an *active mobile adversary* $\mu$ADV [92]. $\mu$ADV is not only able to eavesdrop the communications between any two nodes of the network, but can even corrupt a number of nodes and gain access to all the data and key material they store. However, we make some conservative assumptions to limit the impact of such a powerful adversary. Firstly, we assume that a secure symmetric key protocol is implemented for node-to-node communications [122] – hence, sensed data are encrypted – so that the adversary can only access data received and stored by the corrupted nodes, for which it has access to the key. Secondly, since in our scheme data are only exchanged locally, we assume that the effect of traffic analysis is negligible. In particular, the adversary has no access to even contextual information about shares stored by non corrupted nodes – even if such shares are exchanged using wireless communication. Thirdly, and most importantly, we assume that the number of contemporaneously corrupted nodes is a small fraction of the number of nodes accessed by the sink in a data gathering operation. Indeed, if the amount of sensed, plaintext, data available to the adversary is the same available to the sink, *no scheme* can provide both availability and confidentiality. A similar assumption relies on advanced functionalities like healing schemes [104] (able to recover previously captured nodes) and proactive secret sharing [134] (used to periodically update the shares). Such primitives, supported by the inability of the adversary to operate continuously and/or on the whole network to elude detection, allow to assume that only data stored by recently corrupted sensors are actually accessible to any attacker.

### 3.1.3 Design Goals

The purpose of our analysis is to show that local information sharing, combined with nodes mobility, can: (i) provide an optimal trade-off between data availability and confidentiality, (ii) enhance the level of protection against fake-data injection and information traceability, improving data integrity and source-location privacy, and (iii) avoid heavy computations and energy-expensive routing protocols. To enforce data *availability* means to minimize the amount of data lost as a consequence of nodes failure or capture. To provide data *confidentiality* means to avoid that stored data are exposed to unauthorized entities. These two security requirements are usually in conflict, because availability can only be ensured introducing some sort of redundancy, which often ends up exposing confidential data. To address data *integrity*, means to ensure that the data reconstructed by the sink coincide with those originally sensed by the nodes, and were not modified by the attacker. Finally, with *source-location privacy* we refer to concealing all information related to the origin of the sensed data. On the whole, our analysis wants to stress the positive impact of information diffusion on all the former security requirements. In unattended networks, it must be conciliated with energy saving, and we aim at showing that this is feasible leveraging secret sharing and nodes mobility. However, our main goal is to precisely describe how a proper adjustment of system parameters can concurrently optimize the data recovery process and minimize the risk of information leakage. To this end, we will state a (probabilistic) lower bound on the number of secrets that the sink can reconstruct, based on the total amount of shares it is able to access, and an analogous upper bound for the adversary.

## 3.2 Scheme Description

We are now ready to formally describe our scheme. After a recap of threshold secret sharing schemes (Section 3.2.1), we will exhibit our proposal based on an efficient implementation of information sharing and diffusion (Section 3.2.2).

### 3.2.1 Preliminaries: Secret Sharing

Assume a user $u$ knows a secret $D$. A $(k, n)$-*threshold secret sharing scheme* allows $u$ to choose two positive integers $n$ and $k \leq n$ and to generate $n$ pieces of information, such that any $k$ out of them are necessary and sufficient to recover $D$. Sharing schemes were formally introduced independently by Shamir [129] and Blakley [135]. We will use Shamir's definition, based on polynomial interpolation and readily applicable, but the two schemes are *de facto* equivalent. Assume the secret is represented as an element of some finite field $\mathbb{F}_q$. Shamir's scheme relies on the secrecy of the polynomial of degree $k - 1$

$$f(x) = D + \alpha_1 x + \cdots + \alpha_{k-1} x^{k-1} \in \mathbb{F}_q[X]$$

in which the free term is set equal to $D$, while the $k - 1$ coefficients $\alpha_1, \ldots, \alpha_{k-1}$ are picked uniformly at random in $\mathbb{F}_q$. Knowing any $k$ points of the curve defined by $f(x)$, anyone can recover $f(x)$, and thus $D$. However, given any $k' < k$ points, the uncertainty about both $f(x)$ and $D$ is maximal. The $n$ pieces of information $d_1, \ldots, d_n$, denoted *shares*, are therefore defined as couples $d_j = (d_{j,1}, d_{j,2})$, such that $d_{j,2} = f(d_{j,1})$. The first component $d_{j,1}$ can be defined implicitly and omitted[1], so that each share has the same size of the secret $D$, making Shamir's scheme a so-called *minimal* scheme. The scheme is also *perfect*: with less than $k$ shares, anyone of the possible $q$ values for $D$ is exactly equally likely, and no information about $D$ is leaked. A minimal and perfect scheme like Shamir's is called an *ideal* scheme. The user $u$ that generates and distributes the shares is called the *dealer*.

[1]In our scenario, for instance, the share $d_j$ destined to sensor $s_{i_j}$ can be defined using the unique index $i_j$ as $d_{j,1} = i_j$.

### 3.2.2 Scheme Details

In our scheme, whenever a node $s$ senses some relevant data $D$, it acts as a dealer of the depicted $(k, n)$ threshold secret sharing scheme. Including itself among the participants, $s$ distributes $n - 1$ shares to as many randomly picked neighboring nodes $s_{i_1}, \ldots, s_{i_{n-1}}$, and keeps the last one for itself. Eventually, $s$ securely deletes $D$, which is then shared among $s, s_{i_1}, \ldots, s_{i_{n-1}}$. In Section 5.4.2, we denoted with $D(i, t) \in \mathbb{F}_q$ the data sensed by node $s_i$ at a given time $t$. Since our analysis only focuses on the time window between $t$ and the arrival of the sink or the adversary, in the remainder of the chapter we will ease the notation and refer $D(i, t)$ as $D(i)$. Formally, our scheme consists of the following steps, executed independently by each sensor $s_i$ in each round:

**Shares Generation** $s_i$ applies a $(k, n)$ secret sharing scheme to the sensed information $D(i) \in \mathbb{F}_q$, obtaining the $n$ shares $d_1(i), \ldots, d_n(i)$.

**Local Distribution** $s_i$ randomly selects $n - 1$ neighboring nodes $s_{i_1}, \ldots, s_{i_{n-1}}$, sends $d_j(i)$ to $s_{i_j}$, saves $d_n(i)$ in its own storage facilities, and securely deletes $D(i)$.

**Information Diffusion** Each secret $D(i)$ is now shared among the nodes $s_i, s_{i_1}, \ldots, s_{i_{n-1}}$, which means that all nodes now store shares received by different neighboring sensors. Using sensors random movements as a diffusion medium, the sensed information gets spatially spread in $A$.

**Data Collection** The sink starts exploring $A$ at time $t + \tau_S$. As a node enters the sink's communication range, all the shares the node stores are offloaded to the sink and promptly deleted from the memory of the sensor. As soon as it meets a predetermined number of sensors $m_S$, the sink leaves the network.

*Remark* 3.2.1 (Behavior of the adversary). We assume that the adversary starts exploring $A$ at time $t + \tau_A$, trying to corrupt as many nodes as possible. However, as discussed in Section 3.1.2, the adversary must leave the network quickly to evade detection, and can corrupt a number of nodes $m_A$ remarkably smaller than $m_S$. Relying on primitives like healing schemes [104] or proactive secret sharing [134], we assume that only data stored by the $m_A$ newly corrupted sensors are accessible to the adversary.

*Remark* 3.2.2 (Connectivity). Our scheme implicitly requires $(n - 1)$-connectivity: each node needs at least $n - 1$ neighbors to share the sensed information with. The precise configuration of the network at any specific time is unpredictable, but the topology of the network can be modeled as a *random geometric graph* [136], whose characteristics depend on the density and communication range of the nodes. In particular, for any arbitrarily small $\epsilon$, it is possible to identify the largest $n$ ensuring $(n-1)$-connectivity with probability at least $1-\epsilon$. Indeed, it holds [137]

$$\Pr[\deg_{\min} \geq n_0] = \left(1 - \sum_{i=0}^{n_0-1} \frac{(\rho B)^i}{i!} e^{-\rho B}\right)^N \tag{3.1}$$

where $\deg_{\min}$ denotes the minimum degree of the network, $N$ the total number of nodes, $\rho$ the node density, and $B$ the average communication area of the nodes.[2] Consequently, we can define

$$n_{\max} = \max\{n_0 : \Pr[\deg_{\min} \geq n_0] > 1 - \epsilon\} \tag{3.2}$$

and impose $n = n_{\max} + 1$.

In the unlucky circumstance that $\deg(s_i) < n-1$ for some sensor $s_i$, $s_i$ can ask some neighbors to *route* the exceeding shares to their own neighbors. Using a similar approach does not affect the total number of messages sent by the network, as long as multiple shares can be included in the same message. It just causes a slight imbalance in the workload of the sensors, because routers must compensate for the minor number of messages sent by $s_i$. Fig. 3.1 shows how $n_{\max}$ varies, based on network size $N$ and communication range $r$. We assume that the monitored area is the $[0,1] \times [0,1]$ unitary square, and, assuming constant density to simplify, we deduce it from $N$.Instead of fixing an absolute threshold $\epsilon$, we let it vary with $N$ as $\epsilon = 1/20N$.[3] The function plotted in Fig. 3.1 is

$$n_{\max} = \max\{n_0 : \Pr[\deg_{\min} \geq n_0] > 1 - 1/20N\}$$

as $r$ and $N$ vary. In particular, we invite the reader to observe (it will be useful later) that when $N = 500$, $r = 0.15$ implies $n_{\max} = 4$ and $r = 0.2$ implies $n_{\max} = 9$.

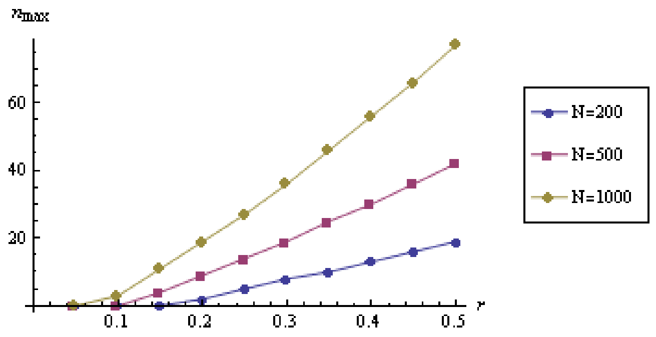


FIGURE 3.1: Minimum connectivity of the network.

We implemented the formerly described "route-when-necessary" strategy in all of our 72000 simulations (discussed in Section 3.6). The network was composed of $N = 500$ nodes, and

[2]If $A$ has a *toroidal* topology (no "border-effects"), and $r$ denotes the communication range of the nodes, it simply holds $B = \pi r^2$.

[3]In other words, we are are bounding to $1/20 < 1$ (an arbitrary value) the expected number of nodes which need routing to deliver the shares.

$n_{\max}$ was set according to (3.2), with $\epsilon = 10^{-4}$. Table 3.1 summarizes how much routing was needed to complete the shares distribution: as expected, *routing* was de facto *non-necessary* or negligible.

| No. of extra messages | 0 | 1 | 2 | 3 | > 3 |
|---|---|---|---|---|---|
| Times they were necessary | 71006 | 682 | 262 | 43 | 7 |
| Perc. of the simulations | 98.72% | 0.95% | 0.36% | 0.06% | 0.01% |

Table 3.1: Required Routing in Shares Distribution

Finally, it is worth to mention a simpler and more effective strategy. As already observed in the literature [111], mobility can be leveraged to improve the connectivity of the network. Along the same line, the requirement of $(n-1)$-connectivity can be relaxed, by simply establishing that the shares are not distributed instantly, but each sensor can wait until meets at least $n-1$ different nodes. This way, the scheme can be adapted to any kind of mobility model without ever requiring routing of the shares. However, to deal with a more predictable sharing process and simplify the analysis, we will not discuss this possibility any further.

## 3.3 Predicting Information Recovery in Sharing Protocols

In this section, we present the core of our work: a thorough analysis of the information-recovery process, able to describe the probabilistic dependence of the amount of information reconstructed on the number of sensors accessed. To enhance readability, we separately expose the main results (Section 3.3.1) and the proofs (Section 3.3.2).

### 3.3.1 Main Results

In the following, let $\text{Poi}_\lambda(E)$ and $\text{Bin}_{l,p}(E)$ denote the probability of event $E$ for a Poisson of mean $\lambda$ and a Binomial of parameters $l$ and $p$, respectively.

**Theorem 3.3.1.** *Assume each sensor $s_i$ shares the secret $D(i)$ with other $n-1$ randomly selected neighbors, as described in Section 3.2. Let us pick $m$ nodes of the network uniformly at random, and define the random variable* $\text{Rec}(m)$ *counting the number of secrets that can be reconstructed only based on the shares stored by such $m$ sensors. The following probabilistic bounds hold:*

$$\Pr[\text{Rec}(m) \le h] \le 2\text{Bin}_{N,p(m)}([0,h]) \tag{3.3}$$

$$\Pr[\text{Rec}(m) \ge h] \le 2\text{Bin}_{N,p(m)}([h,N]) \tag{3.4}$$

*where $\mu(m) = mn/N$ denotes the expected number of shares collected for any $D(i)$, and $p(m) =$* $\mathrm{Poi}_{\mu(m)}([k, +\infty))$.

□

**Corollary 3.3.2.** *Assume that the sink gathers all the shares stored by $m_S$ randomly selected nodes of the network. Let $\mathrm{Rec}(m_S)$ denote the random variable counting the number of secret sensed data that the sink can reconstruct only based on the shares stored by such $m_S$ sensors. Then:*

$$\Pr[\mathrm{Rec}(m_S) \leq h] \leq 2\mathrm{Bin}_{N,p(m_S)}([0, h]) \tag{3.5}$$

*As a consequence, for all $h < Np(m_S)$,*

$$\Pr[\mathrm{Rec}(m_S) \leq h] \leq 2\left(\frac{Np(m_S)}{h}\right)^h e^{-(Np(m_S)-h)} \tag{3.6}$$

□

**Corollary 3.3.3.** *Assume that the adversary gathers all the shares stored by $m_A$ randomly selected nodes of the network. Let $\mathrm{Rec}(m_A)$ denote the random variable counting the number of secret sensed data that the sink can reconstruct only based on the shares stored by such $m_A$ sensors. Then:*

$$\Pr[\mathrm{Rec}(m_A) \geq h] \leq 2\mathrm{Bin}_{N,p(m_A)}([h, N]) \tag{3.7}$$

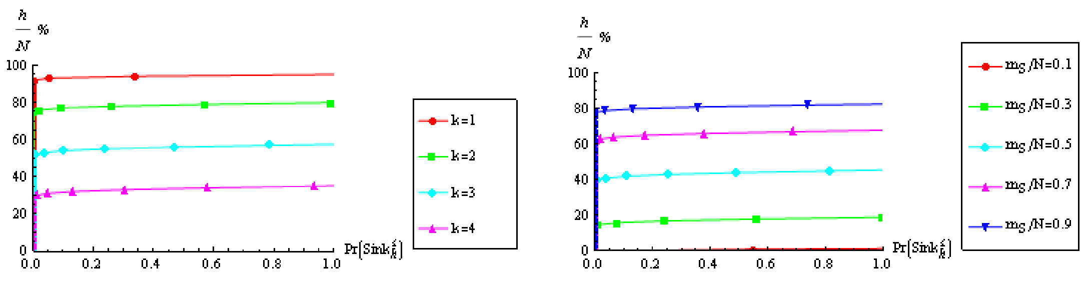
*As a consequence, for all $h > Np(m_A)$,*

$$\Pr[\mathrm{Rec}(m_A) \geq h] \leq 2\left(\frac{Np(m_A)}{h}\right)^h e^{-(h-Np(m_A))} \tag{3.8}$$

□

(3.6) and (3.8) provide weaker bounds than (3.5) and (3.7), but they show more explicitly the exponential drop of the tails $\Pr[\mathrm{Rec}(m_S) \leq h]$ and $\Pr[Rec(m_A) \geq h]$. To permit a better understanding of (3.5) and (3.7), we plot such bounds in Figs. 3.2 and 3.3, letting the parameters $k$, $m_S$ and $m_A$ vary. The size of the network and the communication range of the sensors are set as $N = 500$ and $r = 0.15$, respectively. Consequently (see Remark 3.2.2), we impose $n = n_{\max} + 1 = 5$. In Figs. 3.2a and 3.3a, we fix $m_S/N = 0.6$ and $m_A/N = 0.1$, and compare the bounds for different values of $k$. In Figs. 3.2b and 3.3b we do the opposite, fixing $k = 3$ and letting $m_S/N$ and $m_A/N$ vary.

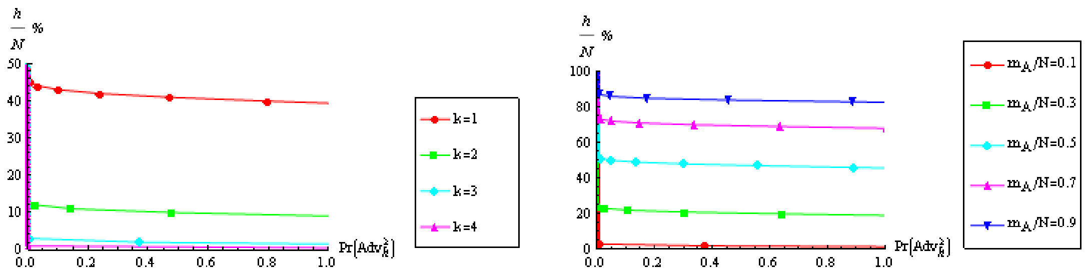
By a look at the plots, we observe that: (i) the proposed scheme is very effective, with respect to a scenario where each sensor keeps its own data, and (ii) the consequences of statistical fluctuations are negligible, with the distribution of the number of recovered secrets being sharply *concentrated* around its expected (and most probable) value. To better motivate the latter statements, let us momentarily focus on the case of a network using a $(3, 5)$ secret sharing scheme. Fig. 3.2a shows that, accessing 60% of the network, the sink can almost surely reconstruct a percentage between 50% and 60% of the sensed data. Similarly, Fig. 3.3a shows that, corrupting 10% of the network, the adversary can almost surely reconstruct a percentage between 2% and 3% of the sensed data. If each sensor kept its own data, the percentage of sensed data collected would simply equal the accessed (or corrupted) percentage of the network. In other words, the effects of a $(3, 5)$ scheme are not only highly predictable, but also highly desirable: while the sink's performances only slightly degrade (the expected percentage of information recovered decreases from 60% to approximately 55%), the consequences of an attack are reduced of approximately 75% (the expected percentage of information stolen decreases from 10% to approximately 2.5%). Summing up, the scheme provides a remarkable trade-off between confidentiality and availability.



(a) $m_S/N = 0.6$, varying $k$.

(b) $k = 3$, varying $m_S/N$.

FIGURE 3.2: Percentage of data recovered by the sink, according to (3.5).



(a) $m_A/N = 0.1$, varying $k$ (for clarity, only $h/N \in (0, 1/2)$ is shown).

(b) $k = 3$, varying $m_A/N$.

FIGURE 3.3: Percentage of data recovered by the adversary, according to (3.7).

### 3.3.2 Proofs

In this section, we will prove Theorem 3.3.1, Corollary 3.3.2 and Corollary 3.3.3. The proof of Theorem 3.3.1 relies on a deeper analysis of the information-recovery process, mostly inspired by the well-known *balls into bins* process and by the work [138] by Mitzenmacher and Upfal.

Corollaries 3.3.2 and 3.3.3 follow easily from Theorem 3.3.1 and from well known tail bounds from the literature.

Before going through the details of the proofs, let us state some general considerations. In our scheme, each node sends its shares to $n-1$ randomly chosen neighbors and keeps the last share for itself. Since sensors are randomly deployed, if $N$ is sufficiently large, the number of shares stored by each node is sharply concentrated around the expected value $n$. Indeed, a Chernoff-like bound for the number of shares stored by each node could be easily stated, leveraging that sensors pick the addressees of the shares uniformly at random, and independently from each other. For the sake of simplicity, our theoretical analysis relies on the assumption that every node $s_i$ stores *exactly* $n$ shares. Particularly since we analyze events that involve a large number of nodes altogether, we are confident that this assumption approximates the reality sufficiently precisely not to invalidate our results. The simulations presented in Section 3.6 will prove us right.

Summing up, our model can be described as follows:

- When the sink or the adversary access a node, they obtain $n$ shares originated by $n$ different randomly picked nodes of the network.
- The sink and the adversary select the sensors uniformly at random, therefore there is no correlation between the shares stored by any two of the accessed nodes.

In reality, as well as in our simulations, the sink and the adversary roam in $A$ and retrieve data from the first sensors they meet. In Section 3.4 we will discuss in details in which circumstances the model is realistic and can be used to accurately describe concrete application settings.

#### 3.3.2.1 Proof of Theorem 3.3.1

The shares-recovery process consists in choosing $m$ nodes uniformly at random and gathering all the shares they store. Such a procedure can be modeled by $m$ independent iterations of the following experiment: you start with a set of $n$ balls (corresponding to the $n$ shares stored by a node) and always with the same set of $N$ bins (corresponding to the $N$ nodes of the network); then you throw the balls one by one into a random bin, removing each time the bin where the last thrown ball felt; at the end, the $n$ balls necessarily felt into $n$ different bins (the nodes that generated the corresponding $n$ shares). The two processes only differ by one aspect: in the network, any node generates $n$ shares in total; at the end of the $m$ balls-into-bins experiments, if $m > n$, there can potentially be some bins that contain more than $n$ balls. However, for $N$ much larger than $n$, the event of hitting the same bin more than $n$ times has a negligible probability, hence the balls-into-bins experiments model the process perfectly fine. The relationship between the number $m$ of experiments and the number of bins that contain at least $k$ balls is now a model of the relationship between the number of accessed nodes and the number of recovered secrets.

Let us start with a bit of notation and some initial considerations. We denote $y_i = \{j_1, \dots, j_n\}$ the outcome of the $i$th experiment, meaning that the balls felt into bins $j_1, \dots, j_n$. Accordingly, we write $j \in y_i$ to say that bin $j$ was hit during the $i$th experiment, and define $\mathbb{1}_{j \in y_i}$ as the indicator function of the event $\{j \in y_i\}$. The outcome of each experiment is a randomly chosen selection of the $\binom{N}{n}$ different ways of choosing $n$ bins out of $N$. By construction, every such choice is equally likely and its probability is then $1/\binom{N}{n}$. The probability of a bin $j$ getting hit is equal to the ratio between the number of choices involving that particular bin and the total number of choices. Once a bin is fixed, there remain $N - 1$ bins available and $n - 1$ bins to pick, i.e., $\Pr[\mathbb{1}_{j \in y_i} = 1] = \binom{N-1}{n-1}/\binom{N}{n} = \frac{n}{N}$. Observe that $\mathbb{1}_{j \in y_i}$ and $\mathbb{1}_{j' \in y_i}$ are not independent for any $j \neq j'$. In fact, for the same experiment, knowing that a ball felt into bin $j$ decreases the probability that a ball felt into bin $j'$ as well, and vice versa. To the contrary, the results of different experiments are independent by construction, that is, $\mathbb{1}_{j \in y_i}$ and $\mathbb{1}_{j \in y_{i'}}$ are independent for any $i \neq i'$. Consequently, if we denote $Y_j(m) = \sum_{i=1}^{m} \mathbb{1}_{j \in y_i}$ the variable counting how many balls are in the $j$th bin after $m$ experiments, we know that $Y_j(m) \overset{\mathrm{d}}{\sim} \mathrm{Bin}_{m,n/N}$. If $m$ and $N$ are sufficiently large with respect to $n$ and $h$, it holds $\mathrm{Bin}_{m,n/N} \approx \mathrm{Poi}_{\mu(m)}$, where $\mu(m) = m \cdot n/N$ and where the approximation error is known to be asymptotically bounded by $n/N$ (see, *e.g.*, [139]). It follows that

$$\Pr[Y_j(m) \geq k] \approx \mathrm{Poi}_{\mu(m)}([k, +\infty)) = p(m)$$

The problem when studying the properties of the vector variable $Y(m) = (Y_1(m), \dots, Y_N(m))$, is that, as for the variables $\mathbb{1}_{j \in y_i}$, the components $Y_j(m)$ are not mutually independent. However, in [138], the authors discuss some technicalities that allow to elude the dependence and obtain a very handy bound. Let $\widehat{Y}_1, \dots, \widehat{Y}_N$ be i.i.d. $\mathrm{Poi}_{\mu(m)}$ variables. The distribution of the vector variable $\widehat{Y} = (\widehat{Y}_1, \dots, \widehat{Y}_N)$ conditioned to the event $F = \left\{\sum_{j=1}^{N} \widehat{Y}_j = C\right\}$, where $C$ is a constant, is identical to the distribution of $Y(C)$, regardless of $n$. By studying how the distribution of $\widehat{Y}$ changes when we condition on $F$, the following theorem can be proved:

**Theorem A (5.14 in [138]).** *If $f(z_1, \dots, z_N)$ is a non-negative function such that $\mathbb{E}[f(Y(m))]$ is monotonically increasing or decreasing with the number of experiments m, then*

$$\mathbb{E}[f(Y(m))] \leq 2\mathbb{E}[f(\widehat{Y})]$$

□

Now, take any event $E(z_1, \dots, z_N)$ such that $\Pr[E(Y(m))]$ is monotonically increasing or decreasing with $m$. The most interesting consequence of Theorem A is that $\Pr[E(Y(m))] \leq 2\Pr[E(\widehat{Y})]$, that is, the probability of the event $E$ in the case of the bins-into-balls process is not greater than twice the probability of $E$ in the case of the i.i.d. Poisson variables. In fact, if $f$ is taken as the indicator function of the event $E$, then $f$ is non-negative and $\mathbb{E}[f] = \Pr[E]$ is monotonous, satisfying the hypothesis of the theorem. Theorem A hence provides a way to estimate the relation between accessed nodes and recovered secrets, studying the more manageable case of i.i.d. Poisson variables.

Finally, we can define the indicator functions $\mathbb{1}_{Y_j(m)\geq k}$ and $\mathbb{1}_{\widehat{Y_j}\geq k}$ of the events $\{Y_j(m) \geq k\}$ and $\{\widehat{Y_j} \geq k\}$, respectively. The variable $\mathrm{Rec}(m)$ we defined to count the number of secrets successfully recovered after $m$ nodes have been met can now be expressed as $\mathrm{Rec}(m) = \sum_{j=1}^{N} \mathbb{1}_{Y_j(m)\geq k}$ that is, as the number of bins that, after $m$ experiments, already contain at least $k$ balls. Analogously, for the i.i.d. Poisson variables, let us define the variable $\widehat{Rec} = \sum_{j=1}^{N} \mathbb{1}_{\widehat{Y_j}\geq k}$. First of all, from the linearity of expectation, we know that $\mathbb{E}[\mathrm{Rec}(m)] = Np(m)$. Further, since the number of bins with at least $k$ balls is clearly non decreasing as the number of experiments increases, we know that $\Pr[\mathrm{Rec}(m) \leq h]$ is monotonically decreasing, and $\Pr[\mathrm{Rec}(m) \geq h]$ is monotonically increasing. Consequently, Theorem A provides the bounds:

$$\begin{aligned}\Pr[\mathrm{Rec}(m) \leq h] &\leq 2\Pr[\widehat{\mathrm{Rec}} \leq h]\\ \Pr[\mathrm{Rec}(m) \geq h] &\leq 2\Pr[\widehat{\mathrm{Rec}} \geq h]\end{aligned}$$

Since the variables $\widehat{Y_j}$ are i.i.d. $\mathrm{Poi}_{\mu(m)}$, the indicator functions $\mathbb{1}_{\widehat{Y_j}\geq k}$ are i.i.d. Bernoulli variables of parameter $p(m)$ and their sum $\widehat{\mathrm{Rec}}$ is a $\mathrm{Bin}_{N,p(m)}$. As a consequence, we finally obtain the desired bounds:

$$\begin{aligned}\Pr[\mathrm{Rec}(m) \leq h] &\leq 2\mathrm{Bin}_{N,p(m)}([0,h])\\ \Pr[\mathrm{Rec}(m) \geq h] &\leq 2\mathrm{Bin}_{N,p(m)}([h,N])\end{aligned}$$

#### 3.3.2.2 Proof of Corollary 3.3.2

As we already observed, the position of a node at time $t+\tau_S$ being independent from its position at time $t$, whatever is the strategy used by the sink to roam in $A$ and choose the sensors, it is *de facto* equivalent to a random pick of $m_S$ nodes of the network. This means that we can directly apply Theorem 3.3.1 to obtain the first bound for the sink:

$$\Pr[\mathrm{Rec}(m_S) \leq h] \leq 2\mathrm{Bin}_{N,p(m_S)}([0,h])$$

Now, it only remains to apply the well known (*e.g.*, see [138]) Chernoff's bound for the tail of the Binomial distribution, to get

$$\Pr[\mathrm{Rec}(m_S) \leq h] \leq 2\left(\frac{Np(m_S)}{h}\right)^h e^{-(Np(m_S)-h)}$$

for all $h < Np(m_S) = \mathbb{E}[\mathrm{Rec}(m_S)]$.

#### 3.3.2.3 Proof of Corollary 3.3.3

As for the adversary, similar considerations to those made for the sink are valid: if the position of each node at time $t + \tau_A$ is independent from its position at time $t$, any line of attack is equivalent to corrupting $m_A$ randomly picked sensors. Once again, we can directly apply Theorem 3.3.1 to obtain the first bound for the adversary:

$$\Pr[\mathrm{Rec}(m_A) \geq h] \leq 2\mathrm{Bin}_{N,p(m_A)}([h, N])$$

Again, using Chernoff's bound for the tail of the Binomial distribution, we finally get

$$\Pr[\mathrm{Rec}(m_A) \geq h] \leq 2\left(\frac{Np(m_A)}{h}\right)^h e^{-(h-Np(m_A))}$$

for all $h > Np(m_A) = \mathbb{E}[\mathrm{Rec}(m_A)]$.

## 3.4 Can We Predict the Effects of Random Mobility?

In DSNs, sensors can move according to very different motions. The way they explore the environment is a very important parameter, which affects how (fast) information get diffused. A very flexible and realistic mobility model would be preferable, but it would result very difficult to analyze, preventing any explicit result. As a consequence (as commonly done in the literature with significant results [110–113]), we chose to compare three models, IID, RW and RWP, easy to define and analyze, though able to provide a comprehensive overview of possible application settings. One of the purposes of this section is to formally describe the properties of these three models, using the apparatus of notations and results of the theory of stochastic processes (Section 3.4.1). However, we will subsequently (Section 3.4.2) show that to study the properties of the mobility model is not enough: on their own, they cannot capture how other parameters, which describe the processes of information sharing and recovery, affect the time required for the performances of the network to become predictable.

### 3.4.1 Mobility Models Are Stochastic Processes

A *stochastic process* is a family of random variables $\{X_t, t \in T \subset \mathbb{R}^+\}$, indexed by a parameter $t$ (usually denoting time), defined over the same sample space $\Omega$, and taking values in a set $S$ called *state space* of the process. A stochastic process is called a *Markov process* if it has "no memory": given $X_t = x$, the transition probability $P(X_{t'} = x')$ for all $t' > t$ does not depend on the history of the process, but only on $x$ (and possibly $t$). In our scenario, the position at time $t$ of a sensor $s$ can be modeled as a Markov process $X_t$ taking values in $A$, to which a probability distribution $\nu_t$ can be associated. When the state space $S$ is finite (*i.e.*, nodes can be located only in a finite number of points of $A$), $\nu_t$ can be represented as a vector of dimension $|S|$, where for every $x \in S$ the coordinate $\nu_t(x)$ is the probability $\Pr[X_t = x]$. Otherwise, the probability distribution is expressed through a probability density function $\phi_t(x)$ such that, for any $W \subset S$, it holds $\nu_t(W) = \int_W \phi_t(x)dx$. Under suitable conditions [140], verified by any reasonable mobility model, a Markov process admits a limiting distribution $\nu$, such that $\lim_{t\to\infty} \nu_t = \nu$. $\nu$ is usually referred to as the *stationary distribution* of the process, since, once the process reaches the distribution $\nu$, it keeps the same distribution thereafter. The *mixing time* of a Markov process measures the time necessary for the distribution of a process to get sufficiently close to stationarity. Given a distance $d$ for probability measures, and an arbitrary constant $\epsilon < 1/2$, the mixing time $t_{\text{mix}}(\epsilon)$ of the process $X_t$ is defined as $t_{\text{mix}}(\epsilon) := \min\{t > 0 : \ d(\nu_t, \nu) \leq \epsilon\}$. It can be shown that the order of magnitude of the mixing time does not depend on $\epsilon$, thus it is normal practice to use $t_{\text{mix}}(1/4)$, usually denoted just as $t_{\text{mix}}$. Observe that, since the stationary distribution is independent from the initial conditions, the mixing time can be equivalently seen as a measure of the time necessary for $X_t$ to lose correlation with its starting location $X_0$.[4]

In short, the stationary distribution tells us how the position of a node will tend to distribute, while the mixing time tells us after how much time the actual distribution will be "close" to the asymptotic one, independently from the initial distribution. IID and RW are two examples of mobility models whose asymptotic probability distribution is *uniform*, but represent the two extreme cases for the time necessary to approach stationarity. In the IID model, by definition, mixing is *instantaneous*, while the RW mixes very slowly: its $t_{\text{mix}}$ is in $O(|A|/l^2 \ln\left(|A|/l^2\right))$ [136], that is, quadratic in the number of states.[5] The RWP, on the other hand, is an example of a mobility model whose stationary distribution is *not uniform*, unbalanced toward the center and almost centrally symmetrical. Its $t_{\text{mix}}$ is a middle ground between those of IID and RW, being in $O(\sqrt{|A|}/l)$ [142].[6] For this reason, the three models considered, even if rather simple in their definition, represent an interesting snapshot of the possible behaviors of the network.

### 3.4.2 What Parameters Do Affect Our Model?

Theorem 3.3.1, presented in section 3.3.1, is the main result of this work. It enables to predict the effects of information sharing, by estimating the amount of sensed data that can be reconstructed only using the shares stored by $m$ sensors. Theorem 3.3.1 relies on two main assumptions that allow us to completely overlook mobility concerns: (i) that $m$ sensors are picked uniformly at

[4]For a complete dissertation on these and similar subjects, we remand the interested reader to [141] or [140].

[5]$|A|/l^2$ is the number of vertexes of a 2D square lattice over $A$, if $l$ is the distance between any two adjacent vertexes.

[6]Here $l = (\nu_{\min} + \nu_{\max})/2$ is the average unitary displacement of a node.

random, and (ii) that to meet a sensor is equivalent to collect $n$ shares generated by $n$ sensors chosen uniformly at random. Our *simplified* model relies on the following rationale: after the sensors moved randomly for *enough* time, spatial constraints are no longer influential, neither for what concerns the way data are gathered, nor for what concerns the way data were shared. In [3], to quantify how much time is *enough*, we proposed to use the mixing time of the mobility model. In fact, the mixing time describes how quickly a sensor loses correlation with its starting position – that is, with the origin of the shares it stores. Unfortunately, the mixing time is unable to capture how other parameters affect the time necessary for our model to precisely describe the real performances of the network, and it often results a too loose upper bound on the time actually needed. In the following, we will briefly highlight the impact of the number $m$ of sensors accessed, of the communication range $r$, and of the total number $n$ of shares generated from any sensed data on the ability to reliably predict the features of the data recovery process. Our purpose for future work is to provide an analysis, even more comprehensive than the one presented in Section 3.3, able to capture the relation among mobility features, time, and system parameters. We are nevertheless confident that the analytical and experimental results presented in this chapter provide a remarkable insight on the settings that allow information sharing and diffusion to yield the best benefits.

In the following, we denote $X_t(i)$ the position of sensor $s_i$ at time $t$. To ease notation, we assume that data were shared at time 0 and are collected at time $\tau$. We also assume that the system already reached stationarity at time 0, meaning that, for all $i \in \{1, \dots, N\}$ and $x \in A$, $\Pr[X_0(i) = x] = \nu(x)$, where $\nu$ is the stationary distribution of the mobility model of the sensors. Observe that, if at time 0 the process already reached stationarity, then the distribution at time $t$ will also be stationary, for all $t > 0$. However, we will sometimes need to analyze the position $X_\tau(j)$ of a sensor $s_j$ at time $\tau$ *with respect to its position* $X_0(j)$ *at time* 0, and to consider how the distribution of $X_\tau(j)$ depends on $X_0(j)$. Finally, we remark that the distribution of the sensors can be seen as a *density*: the number $Z_t(R)$ of sensors in a region $R \subset A$ has expected value

$$\mathbb{E}[Z_t(R)] = N \Pr[X_t(i) \in R] = N\nu_t(R)$$

and the larger is $N$, the more $Z_t(R)$ will be sharply concentrated around its mean.

### Impact of $m$

Assume that $m = N$, that is, the sink (or, equivalently, the adversary) gathers the shares stored by all sensors. In this case, it is clear that the time past from data sensing to data recovery is not relevant. Indeed, the sensors could even not move at all, and all sensed data would be reconstructed anyway. This (extreme) example serves to show that the size of the portion of the network accessed can sensibly affect the impact of nodes mobility. More precisely, in our model we assumed that the set of nodes accessed is equivalent to a selection of $m$ sensors chosen uniformly at random, regardless of the strategy used to explore the network. In other words, we assumed that $\tau$ is large enough to ensure that

$$\max_{\substack{R\subset A:\\ \nu(R)=\frac{m}{N}}} \left| \Pr\left[\forall j = 1,\dots,m:\ X_\tau(j) \in R \mid X_0(j) \overset{\mathrm{d}}{\sim} \nu\right] - \frac{1}{\binom{N}{m}} \right| < \epsilon \tag{3.9}$$

for some sufficiently small $\epsilon$. Indeed, (3.9) says that, if we explore any region $R \subset A$ such that $\nu(R) = m/N$ (*i.e.*, where we expect to find $m$ sensors), for any $m$ sensors whose starting positions $X_0(1),\dots,X_0(m)$ are picked according to $\nu$, the probability to find such $m$ sensors in $R$ is close to the probability of picking them when drawing uniformly at random among the $N$ sensors of the network. One of the possible ways to define $t_{\mathrm{mix}}(\epsilon)$ is as the minimum $t$ such that

$$\max_{R\subset A} \left| \Pr\left[X_t(i) \in R \mid X_0(i) \overset{\mathrm{d}}{\sim} \nu\right] - \nu(R) \right| < \epsilon \tag{3.10}$$

If we compare (3.9) and (3.10), it is easy to realize that, while the latter does not depend on $m$, in the former the size of $R$ grows with $m$, with a twofold effect: (i) the set on which we maximize is reduced, and (ii) as $m$ tends to $N$, the two probabilities tend both to 1, regardless of $\tau$.

### Impact of $r$

Assume that the network enjoys full visibility, so that any sensed data $D(i)$ are shared among $n$ sensors picked uniformly at random among all $N$ sensors. In that case, the hypotheses of Theorem 3.3.1 are clearly immediately verified. In any realistic setting, we cannot assume full visibility, but, intuitively, the larger is $r$, the less correlation exists between the starting position of a node and the shares it stores, hence the faster a local exploration of the network will provide the same result of accessing nodes uniformly at random. Let us try to give a deeper insight. In a sense, we assumed that the probability that a sensor $s_j$ stores a share of $D(i)$ can be approximated by $n/N$, regardless of $i$ and $X_\tau(j)$. Let us denote $B(i) = B_r(X_0(i))$ the area covered by the transceiver of $s_i$ at time 0, and $|B(i)|$ the number of nodes in $B(i)$ (including $s_i$). For the sake of simplicity, let us assume that all $n$ shares $d_{j_1}(i),\dots,d_{j_n}(i)$ are sent to sensors in $B(i)$ picked independently and uniformly at random. Now, the probability that $s_j$ stores one of the $d_{j_h}(i)$ is given by the probability that $s_j$ was in $B(i)$ at time 0, times the probability that $s_j$ was one of the $n$ nodes selected, that is

$$\Pr[X_0(j) \in B(i) \mid X_\tau(j)] \cdot \frac{n}{|B(i)|}$$

We can estimate $|B(i)|$ with its expected value

$$\mathbb{E}[|B(i)|] = N\nu(B(i))$$

Further, by Bayes Theorem we have

$$\Pr[X_0(j) \in B(i) \mid X_\tau(j)] = \Pr[X_\tau(j) \mid X_0(j) \in B(i)] \cdot \frac{\nu(B(i))}{\nu(X_\tau(j))}$$

Putting pieces together, we have

$$\begin{aligned}\Pr[s \text{ stores one of the } d_{j_h}(i)] &= \Pr[X_\tau(j) \mid X_0(j) \in B(i)] \cdot \frac{\nu(B(i))}{\nu(X_\tau(j))} \cdot \frac{n}{N\nu(B(i))} \\ &= \frac{\Pr[X_\tau(j) \mid X_0(j) \in B(i)]}{\nu(X_\tau(j))} \cdot \frac{n}{N}\end{aligned}$$

This makes pretty clearly our point: if $B(i)$ coincides with $A$ (full visibility), the numerator and denominator of the leftmost fraction coincide, because with no information at all on the starting location of $s_j$, its position at time $\tau$ is distributed as $\nu$; in general, the larger is $B(i)$, the less *conditioned* is $X_0(j)$, and consequently the faster the distribution of $X_t(j)$ tends to $\nu$.

### Impact of $n$

To understand the impact of $n$, let us start by a comparison of two extreme cases. First, assume $n = 1$ (and therefore $k = 1$ as well). In this case, no sharing scheme is actually implemented, but data are simply moved to a sensor different from the one that sensed them, similarly to the MOVE-ONCE scheme introduced in [90]. This means that, regardless of how much time elapsed from data sensing to data recovery, and of how far sensors moved in the meanwhile, accessing $m$ sensors allows to collect exactly $m$ sensed data. On the contrary, assume that the network enjoys full visibility (or data are routed) and $n = N$. In this case, each sensor stores a share of each sensed data, and data recovery only depends on how many sensors are met, and not on mobility features. More generally, the larger is $n$, the more likely it is that shares of the same data can be quickly found far away from each other. More precisely, let

$$\nu_t(x) = \Pr[X_t(j) = x \mid X_0(j) \in B(i)]$$

denote the probability that a sensor $s_j$ is in $x$ at time $t$, given that it was in $B(i)$ at time 0. For any region $R \subset A$, we can compute the expected number of shares of data $D(i)$ that can be found in $R$ at time $\tau$ as $n\nu_\tau(R)$, and the larger is $n$, the more likely the actual number will be close to its expected value. Therefore, if we want to find at least $k$ of them, we must impose $n\nu_\tau(R) \geq k$, or equivalently

$$\nu_\tau(R) \geq \frac{k}{n} \tag{3.11}$$

If $R$ is somewhat *far* from $B(i)$, $\nu_t(R)$ is increasing in $t$, for all $t \leq t_{\text{mix}}$. For any fixed $k$, the larger is $n$, the less mixed the network needs to be, hence the smaller $\tau$ can be, for (3.11) to be nevertheless satisfied.

## 3.5 Integrity and Source-Location Privacy Evaluation

In this section, we evaluate how and to which extent the proposed scheme enforces data integrity and source-location privacy. For what concerns the resilience against data poisoning, we will see that the properties of secret sharing itself allow to identify fake shares injected by the adversary, provided that enough legitimate shares are collected (Section 3.5.1). As for source-location privacy, we will instead stress that it is the specific nature of our scheme, which avoids routing combining information sharing and nodes mobility, to make very hard for the adversary to track data back to their source (Section 3.5.2).

### 3.5.1 Data Integrity

Protect data integrity means to defend from fake-data injection, that is, an attack consisting in poisoning the sensed information with arbitrary counterfeit data. To avoid that such fake data are promptly rejected by the sink, without even being considered, the adversary must *authenticate* them, so as to make them look as they were legitimately generated by an authorized node of the network. Consequently, a similar attack does indeed represent a threat only if the adversary injects fake data into the network through some corrupted sensors it is currently controlling. In this context, we need to distinguish between two possible scenarios:

**Fake-Secret Injection (FSeI)** FSeI refers to the possibility that the adversary generates a counterfeit datum $\hat{D}$, and uses a corrupted sensor $\hat{s}$ to inject $\hat{D}$ into the network. $\hat{s}$ behaves as if it actually sensed $\hat{D}$, generating and distributing $n$ shares from the secret $\hat{D}$. Since such shares are obtained exactly as if $\hat{D}$ was indeed sensed by $\hat{s}$, it is *impossible* for the sink to detect that $\hat{D}$ was counterfeit, unless it somehow detects that $\hat{s}$ was corrupted. In other words, it is *impossible* to defend from FSeI, since it would require to be able to distinguish data generated by corrupted and legitimate sensors.

**Fake-Shares Injection (FShI)** FShI refers to the possibility that the adversary uses a corrupted sensors $\hat{s}$ to inject into the network fake shares, which look as they were legitimately generated. In other words, if $\hat{s}$ received a share $d_{\hat{s}}$ of the secret $D$ sensed by an honest sensor $s$, $\hat{s}$ can arbitrarily modify $d_{\hat{s}}$ into $\hat{d}_{\hat{s}}$. This way, if the sink meets $\hat{s}$ after the adversary left the network, the sink will consider $\hat{d}_{\hat{s}}$ as a legitimate share and reconstruct a wrong secret $\hat{D} \neq D$.

Since FSeI cannot be prevented nor tackled, we will only focus on FShI. However, contrarily to FSeI, FShI is not a real threat in many scenarios. In fact, if the shares are signed and/or encrypted with an end-to-end symmetric algorithm before being diffused, a corrupted node cannot modify shares generated by any honest node, without being detected by the sink. However, we will conservatively assume that this is not the case, and discuss what level of protection against FShI can be guaranteed by the implementation of threshold secret sharing.

In Section 3.2.1, we discussed threshold secret sharing schemes, recalling that, for any choice of $k \leq n$, $n$ shares $d_1, \ldots, d_n$ can be generated from a secret $D$, such that any $k$ out of them are necessary and sufficient to recover $D$. Since no information at all about $D$ can be obtained from less than $k$ shares, if the sink is not able to recover at least $k$ of the $d_j$, it will not even try to recover $D$. Therefore, let us assume that the sink recovers $r$ shares $d_{j_1}, \ldots, d_{j_r}$ of the same secret $D$, with $k \leq r \leq n$. If $r = k$, the sink has no other option than using the $k$ shares recovered to reconstruct $D$. In that case, it is sufficient that one of the $k$ shares recovered is counterfeit, to ensure that the sink reconstructs a secret $\hat{D} \neq D$. To the contrary, if $r > k$, the sink knows that the $r$ points corresponding to the $r$ shares recovered should all be on the curve of the polynomial $f(x) = D + \alpha_1 x + \cdots + \alpha_{k-1} x^{k-1}$, which univocally determines the secret $D$. The sink has $\binom{r}{k}$ ways to choose $k$ of the $r$ shares to recover $f(x)$, and thus $D$, from. The recovered secret will be correct if and only if all the $k$ chosen shares are legitimate. Inspired by the latter line of reasoning, we propose the following *secret reconstruction strategy* to optimize the behavior of the sink:

**Detection of fake shares** First, the sink picks any $k$ shares and uses them to reconstruct $f(x)$. Then, it checks each one of the remaining $r - k$ shares against $f(x)$, verifying that the corresponding points satisfy the equation of $f(x)$. If all the shares pass the test, there are only two alternatives: or the shares are all valid, or they are all counterfeit. In both cases, the sink has no other option but accepting the recovered secret as valid, and halts. To the contrary, if there exists at least one share which is inconsistent with $f(x)$, then the sink has detected the presence of at least one fake share, and goes on with the following step.

**Maximum likelihood secret reconstruction** If the sink verified that at least one of the $r$ shares is fake, it can try to probabilistically distinguish between counterfeit and legitimate shares. Since, by hypothesis, the adversary can only corrupt a limited number of nodes, we assume that a share is more likely to be real than fake. Accordingly, following a strategy inspired by the well known *maximum likelihood decoding*, the sink simply recognizes the most probable secret as follows: (i) for each of the $\binom{r}{k}$ subsets of cardinality $k$ of the set $\{d_{j_1}, \ldots, d_{j_r}\}$, the sink reconstruct the secret corresponding to those $k$ shares; (ii) the value that appears more frequently among the $\binom{r}{k}$ results is assigned to the secret $D$ (if there are more than one value appearing with the same, maximum frequency, one of them is picked uniformly at random). Since the values reconstructed using at least one counterfeit share are uniformly distributed over $\mathbb{F}_q$, it can be easily proved that this strategy returns the correct secret $D$, provided that there are at least $k + 1$ legitimate shares, and that there is a majority of legitimate shares in the set $\{d_{j_1}, \ldots, d_{j_r}\}$.

### 3.5.2 Source-Location Privacy

Concerning source-location privacy, the first remarkable improvement provided by our scheme, with respect to protocols relying on data routing for information dissemination, is that *the impact of traffic analysis is completely nullified*, so the protocol is perfectly secure against passive eavesdroppers. In fact, apart from very unlucky circumstances (see Remark 3.2.2 in Section 3.2), no message is ever forwarded, hence there are no data paths that a traffic analysis can highlight.

More generally, we will distinguish the concept of source-location privacy into two different security requirements. Usually, the concept of source-location privacy refers to the level of secrecy concerning the *place* where some data were originated, rather than the identity of the *node* that actually sensed them. In static networks, the two notions are equivalent, because the location of an event automatically identifies the nodes able to observe it. In our context, however, due to nodes mobility, an adversary can recover a piece of information very far from where it was originally generated. As a consequence, we distinguish source-location privacy into:

**Source-Node Privacy (SNP)** SNP, discussed in Section 3.5.2.1, refers to what the adversary can deduce about the *sensor* that generated some intercepted pieces of information.

**Source-Position Privacy (SPP)** SPP, treated in Section 3.5.2.2, refers to what the adversary can deduce about the *place* where the datum corresponding to the intercepted pieces of information was originally sensed.

The remainder of this section relies on the following assumptions. The adversary captured $l$ (with $l \leq n$) sensors storing a share of the same datum $D$. The adversary is able to recognize that those shares were indeed generated from the same secret $D$, and wants to deduce information about the source-node and source-position of $D$. No information about the source of $D$ are accessible in clear-text from less than $k$ shares (e.g., no such information are included in the header of the shares), otherwise source-location privacy would be obviously and fatally compromised. However, other than the sensed datum itself, the secret $D$ also contains all the information about where and by which node that datum was sensed. This means that, if $l \geq k$, the source-node and source-position of $D$ are known to the adversary.

#### 3.5.2.1 Source-Node Privacy

Any sharing protocol, similarly to data replication, introduces at least a basic level of protection with respect to an adversary aiming to identify which node sensed some data $D$: the $n$ nodes carrying the shares of $D$ are indistinguishable from the adversary point of view, providing what is called *n-anonymity*. The purpose of this section is to better quantify the level of source-node privacy provided by a $(k, n)$ threshold sharing scheme, by introducing SNP metrics, depending on $k$, $n$, $N$ and the number $l$ of shares recovered by the adversary.

By hypothesis, if $l \geq k$, the source-node of $D$ is fatally leaked. To the contrary, if $l = 0$, all the $N$ nodes are equally likely the source of $D$. In all middle ground cases, that is, when $0 < l < k$, from the perspective of the attacker, each one of the $l$ corrupted nodes storing one of the shares of $D$ is the source node with probability $1/n$, while any other node is the source node with probability $(n - l)/n(N - l)$. Let $\nu(l)$ be the latter probability distribution over the $N$ sensors. A SNP metric can be naturally induced by any way to measure the level of uncertainty associated to $\nu(l)$.

For instance, we can use Shannon's Entropy [143], (usually denoted with the letter $H$) and define

$$\mathrm{SNP}_H(l, k, n, N) = \sum_{i=1}^{N} Pr[s_i] \log \frac{1}{Pr[s_i]} \quad ,$$

where $Pr[s_i]$ denotes the probability that $s_i$ is the source node. It is straightforward to verify that

$$\mathrm{SNP}_H(l, k, n, N) = \begin{cases} 0 & \text{if } l \geq k \\ \frac{l}{n} \log n + \frac{n-l}{n} \log \frac{n(N-l)}{n-l} & \text{if } 0 < l < k \\ \log N & \text{if } l = 0 \end{cases}$$

Alternatively, the Total Variation (TV) distance [136] between two probability distributions can be used as a metric, comparing $\nu(l)$ to the uniform distribution $\nu(0)$ over the $N$ nodes obtained when $l = 0$. In this case, we can define

$$\mathrm{SNP}_{\mathrm{TV}}(l, k, n, N) = 1 - \|\pi - \nu(l)\|_{\mathrm{TV}} \quad ,$$

and it can be easily verified that

$$\mathrm{SNP}_{\mathrm{TV}}(l, k, n, N) = \begin{cases} \frac{1}{N} & \text{if } l \geq k \\ 1 - l\left(\frac{1}{n} - \frac{1}{N}\right) & \text{if } 0 < l < k \\ 1 & \text{if } l = 0 \end{cases}$$

In both cases, as expected, the SNP is minimized when $l \geq k$, and maximized when $l = 0$.

We assumed that $l \geq k$ gives the adversary complete knowledge about the source-node. If, to the contrary, the source-node is not specified in $D$ (*e.g.*, it may not be interesting for the sink to know the source-node, but just the source-position), the case $l \geq k$ does not represent a special case anymore. In this setting, we would only have, for all $0 \leq l \leq n$,

$$\begin{aligned}\mathrm{SNP}_H(l,k,n,N) &= \frac{l}{n}\log n + \frac{n-l}{n}\log\frac{n(N-l)}{n-l} \quad ,\\ \mathrm{SNP}_{\mathrm{TV}}(l,k,n,N) &= 1 - l\left(\frac{1}{n} - \frac{1}{N}\right) \quad .\end{aligned}$$

Finally, it is important to underline that source-node privacy can be further improved (and be always *maximized*, if $D$ does not contain information on the source-node), upon the requirement of a slightly stronger connectivity, and at the cost of a slightly greater energy consumption. In fact, if the source-node forwards *all* the $n$ shares to as many neighboring sensors, none of the node carrying shares of $D$ actually sensed $D$, and there is absolutely no identifiable correlation between the possession of the shares and their generation.

#### 3.5.2.2 Source-Position Privacy

More realistically, instead of the source-node, the adversary could be interested in identifying the *place* where some data were sensed. The best strategy the attacker can adopt is very well described in [144]. In their model, the network is static and sensors are deployed over a 2D lattice. Whenever a node senses some data, it generates $n$ replica and diffuses them through the network, each replica being independently routed according to a RW on the lattice. The authors of [144] show that the coordinate median of the locations of the intercepted replica provides a good approximation of the source-location, suggesting that the adversary should progressively attack the nodes closer to the identified point.

Apparently, the analysis performed in [144] can be exactly replicated for our scheme, at least if sensors move according to a 2D RW. In fact, the only difference is that the replica (or shares) are carried by mobile nodes, instead of being routed through a static network, but this does not affect the way the information about the source-position is diffused through the network. More generally, under most random mobility models, the idea that the most probable source-position of a sensed datum is the coordinate median of the points where the corresponding shares were recovered may sound extremely reasonable. However, such an insight, which can be theoretically and empirically proved under the hypothesis that the shares did not get *too far* from their starting location, turns out as completely misleading when such an assumption is not satisfied.

To understand why this it true, we can just think to a very important property of all random processes: the mixing time (see Section 3.4). As soon as the time elapsed from the shares distribution till the nodes capture is of the same order of magnitude of the mixing time of the diffusion model, by definition, there is only a very small correlation between the positions of the nodes carrying the shares and the position where the shares were generated. Consequently, *any* *possible source-position is (almost) equally likely*, no matter where the shares are found. The analysis in [144] makes sense only because, in static networks, the $n$ replicas are necessarily routed for a predetermined small number of hops, due to the constraint that the length of the routing path must be traded-off against energy saving. In our scenario, to the contrary, the shares are carried by continuously moving nodes, whose position will eventually become independent from where data were sensed, making all probabilistic considerations about the source-position meaningless. A similar attack can be applied to our scenario only for *fresh* data, that is, data generated sufficiently recently. For all the so-called *historical data*, the privacy of the place where data were sensed is *maximal*, no matter what strategy the adversary adopts.

According to the former line of reasoning, we can introduce the following metric for the SPP. We define $\nu$ the stationary distribution of the mobility model of the network, and $\nu_t(x)$ the probability distribution at time $t$ of the position of a node which was in point $x \in A$ at time zero. If $x_D$ is the place where $D$ was sensed, we define

$$\mathrm{SPP}(t) = 1 - \|\nu - \nu_t(x_D)\|_{\mathrm{TV}}$$

as the complementary of the TV distance between $\nu_t(x_D)$ and $\nu$. When $t = 0$, the distance between $\nu_t(x_D)$ and $\nu$ is maximized, so the SPP is minimized, according to the idea that intercepting a share when it was just generated gives complete information about the source-position. To the contrary, the more time passes, the more $\nu_t(x_D)$ will tend to $\nu$, so the SPP will tend to 1, according to the idea that after a sufficiently large time the uncertainty about the source-position is maximal. Summing up, as expected, the SPP strongly depends on the relation between the time past from the shares generation and the level of mobility of the network.

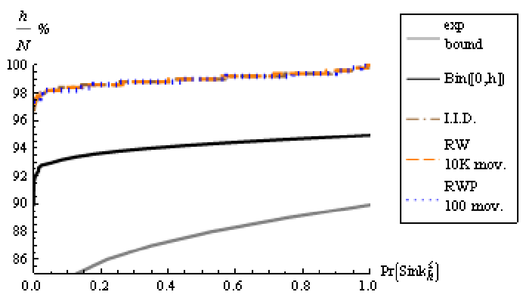
## 3.6 Simulations and Discussions

In this section, we show and discuss some plots describing the information recovery phase. First, we compare theoretical and experimental plots to further validate the analytical bounds presented in Section 3.3. Then, we show how the recovery phase is affected by information diffusion, by comparing the experimental plots obtained for different values of the time interval $\tau$ between the distribution and the recovery of the shares.

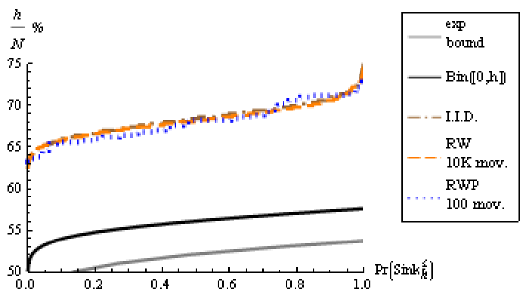
The simulations involve the entire process of information diffusion and data recovery and every plot is obtained averaging the results of 1000 independent simulations. The experimental settings can be summed up as follows. $N = 500$ nodes are randomly deployed in the unitary square $[0, 1] \times [0, 1]$, that simulates the monitored area $A$. Sensors' communication range is set as $r = 0.15$ and, according to Remark 3.2.2 in Section 3.2, we fix $n = 5$. We compare the replication scheme, which corresponds to setting $k = 1$, with the two threshold schemes corresponding to $k = 3$ and $k = 4$. Assuming each sensor acquired some data $D$ at time $t$, we simulate the random distribution of the shares among the neighbors. Then, the nodes start moving according to one of the three mobility models proposed. For the RW, we set $l = 0.01$, so that $A$ contains a grid composed by 10000 vertices. For the RWP, we set $v_{\min} = 0.005$ and $v_{\max} = 0.015$, so that the average unitary displacement of a node is again $l = 0.01$. At time $t + \tau$, the sink, or analogously the adversary, explores the network with rectilinear movements, each one from a randomly chosen access point on the border of $A$ to a randomly chosen exit point. They stop when they meet a fixed fraction of the network, that is the 60% (300 nodes) for the sink and the 10% (50 nodes) for the adversary, and we compute the number of secrets they successfully recover.

Figs. 3.4 and 3.5 show a comparison between the upper bounds (3.5), (3.6), (3.7) and (3.8) and the empirical plots for the three mobility models. $\tau$ is set of the same order of magnitude of its mixing time for the RWP, while we set $\tau = |S|$ ($S$ is the state space of the process, see Section 3.4.1) for the RW, *i.e.*, a time interval actually *shorter* than the known bound on its mixing time. What can be observed is that the amount of secrets recovered by the sink is remarkably *larger* than what (3.5) suggests for both $k = 1$ (Fig. 3.4a) and $k = 3$ (Fig. 3.4b), while the amount of secrets recovered by the adversary is remarkably *smaller* than what (3.7) suggests for both $k = 3$ (Fig. 3.5b) and $k = 4$ (Fig. 3.5c). In the remaining two cases, we observe a slightly different behavior. The rationale is that the two extreme cases $k \approx n$ and $k \ll n$ are more susceptible to the gap between the theoretical model and the experimental setting, and the consequences emerge in particular when we try to lower bound the amount of information recovered if $k$ is large, and when we try to upper bound it if $k$ is small. Nevertheless, our bounds (3.5) and (3.7) still provide a very tight approximation, while the exponential bounds (3.6) and (3.8) remain valid.

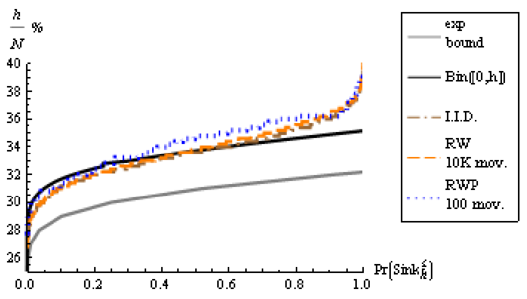
In Figs. 3.6 and 3.7 for the RW and Figs. 3.8 and 3.9 for the RWP, we analyze how time affects the information-recovery process. The plots show exactly what we expected. Firstly, as long as $\tau$ is small, the process shows high variance, since it is very unpredictable whether or not the nodes are already well spread into $A$. Secondly, information diffusion generally improves both availability and confidentiality of the sensed data. The insight is that, as long as the shares are locally concentrated, the sink cannot find enough shares of data sensed in farther regions of the monitored area. To the contrary, if information is not diffused, the adversary can easily find enough shares of the same secrets though corrupting only a small number of nodes. The only two cases that show a different behavior concern the sink when $k = 4$, and the adversary $k = 1$. In this case, the rationale is that, when $k = 4$, the task of recovering 4 out of 5 shares of any secret becomes difficult for the sink as well, and it can profit from low diffusion (Figs. 3.6c and 3.8c). On the other hand, when only 1 share is needed, capturing many nodes which carry the same shares turns into an useless effort, which is particularly relevant for an adversary that can only capture a limited amount of sensors (Figs. 3.7a and 3.9a). Summing up, the analysis of



(a) $k = 1$.

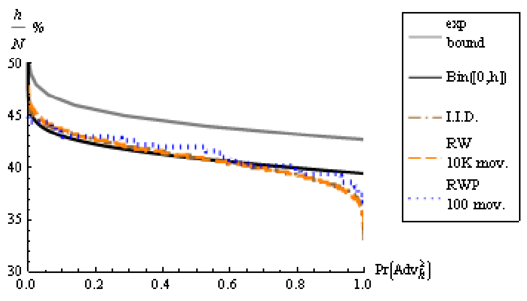


(b) $k = 3$.

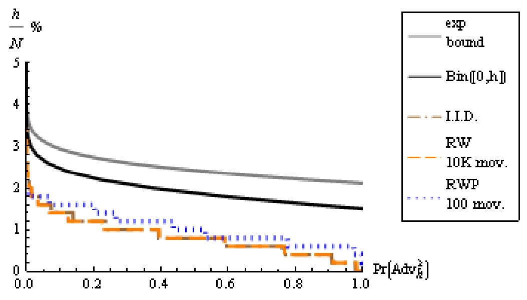


(c) $k = 4$.

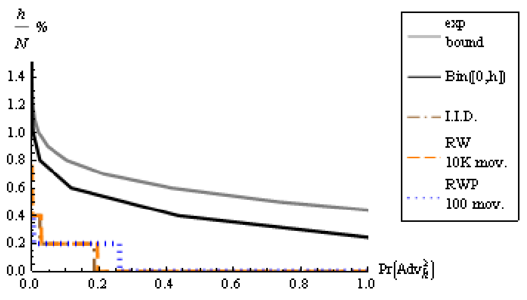
FIGURE 3.4: Data recovery process for the sink meeting the 60% of the network: theoretical vs. empirical comparison.



(a) $k = 1$.



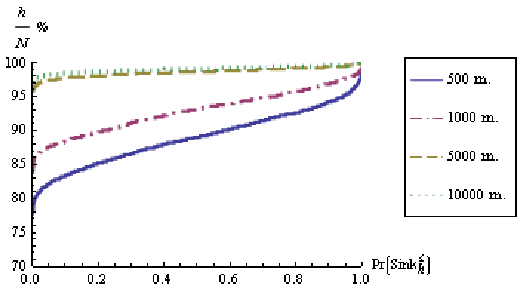
(b) $k = 3$.



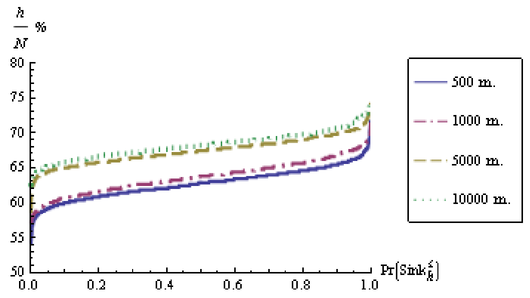
(c) $k = 4$.

FIGURE 3.5: Data recovery process for an adversary capturing the 10% of the network: theoretical vs. empirical comparison.

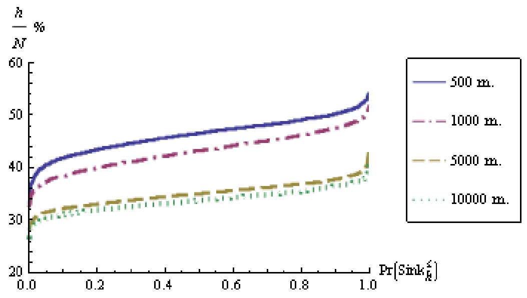
the time impact underlines that the choice of the relation between the parameters $k$ and $n$ should strongly depend on the mobility level of the network.



(a) $k$ = 1.



(b) $k$ = 3.

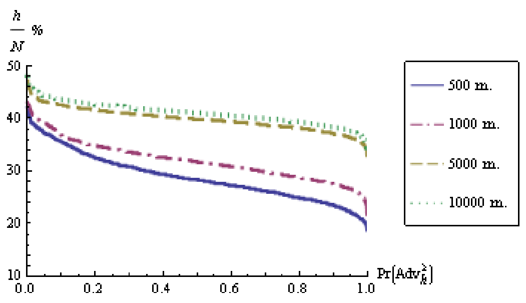


(c) $k$ = 4.

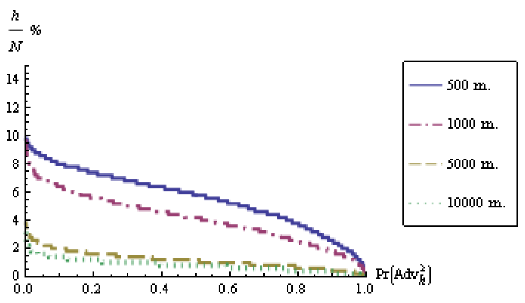
FIGURE 3.6: Time impact on the data recovery process for the sink meeting the 60% of the network, under the RW model.

## 3.7 Summary

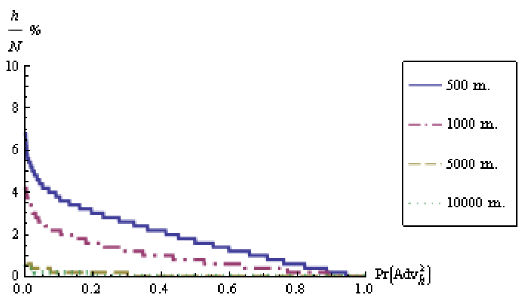
The work presented in this chapter provides a novel approach to information security in Distributed Sensor Networks (DSNs). We showed that a secure and efficient data handling scheme can be realized simply based on a local implementation of secret sharing, and leveraging sensors mobility to diffuse the pieces of information generated. The rationale was to make the content of information shared by the nodes somehow independent from their position, so that the amount of data that can be recovered accessing a fraction of the network only depends on the size of that fraction. We bounded the amount of information retrievable by the sink and by the adversary, as a function of the parameters $k$ and $n$ of the secret sharing scheme and of the number of accessed nodes. This way, we proved that secret sharing and shares spatial diffusion allow to penalize adversaries that can only capture a small number of nodes, while favoring an intermittent sink that sporadically explores a significant portion of the monitored area. In other words, our work permits to tune the parameters of the scheme so as to obtain the desired trade-off between availability and confidentiality of the sensed data, which are the two primary concerns in DSNs. Further, we empirically showed the importance of the time past between data sensing and collection, and how its impact can be predicted based on the characteristics of the mobility model the



(a) $k = 1$.

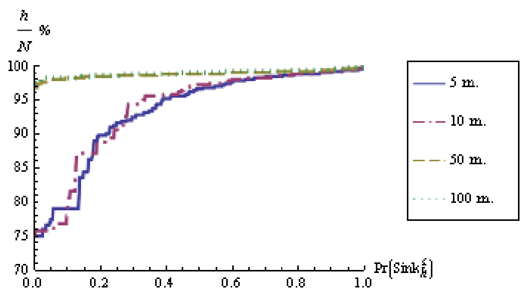


(b) $k = 3$.

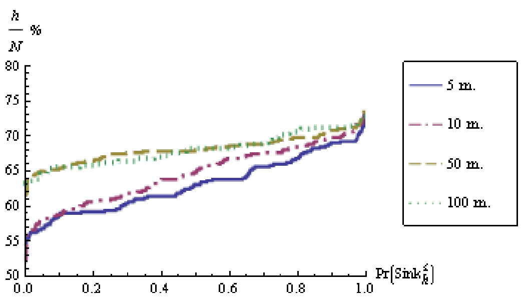


(c) $k = 4$.

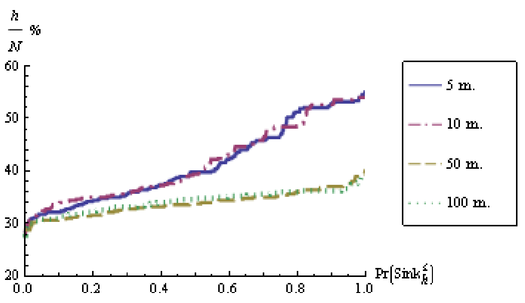
FIGURE 3.7: Time impact on the data recovery process for an adversary capturing the 10% of the network, under the RW model.



(a) $k = 1$.

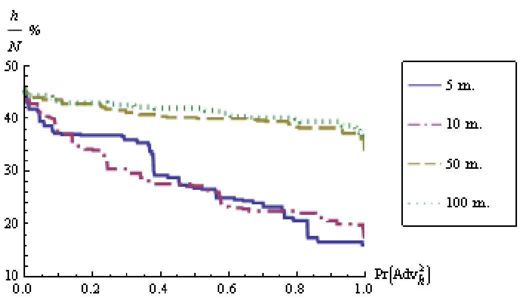


(b) $k = 3$.

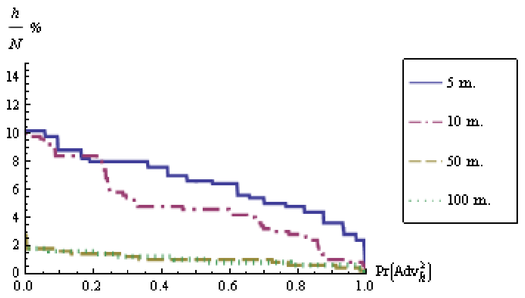


(c) $k = 4$.

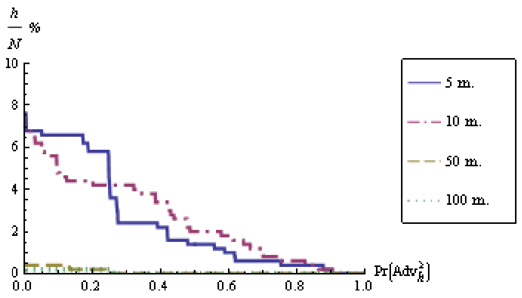
FIGURE 3.8: Time impact on the data recovery process for the sink meeting the 60% of the network, under the RWP model.



(a) $k = 1$.



(b) $k = 3$.



(c) $k = 4$.

FIGURE 3.9: Time impact on the data recovery process for an adversary capturing the 10% of the network, under the RWP model.

nodes of the network are subject to. However, we underlined that other parameters (primarily $n$, the sensors communication range $r$, and the number $m$ of sensors accessed by the sink) can affect the speed to which information "mixes", and must therefore be taken into account when analyzing the performances of the system. Finally, we discussed how our scheme enhance the level of protection against fake-data injection and data traceability, introducing specific metrics for the data integrity and source-location privacy. Extensive simulation results supported all of our above findings.

# Part II

# Provable Storage Medium for Cloud Storage Systems

# 4 Storing Data on the Cloud: Security and Privacy Issues

In modern network and computing applications, a common practice is to move services to the cloud. Recent technologies indeed offer to both common users and big companies the possibility to rely on external cloud servers for the outsourcing of both data storage and onerous computations on such data, on a pay-per-use base. Motivations for companies and private users to rely on these services vary from increased service availability, to higher reliability, lower maintenance cost, etc. [145]. Further, cloud computing often allows to perform operations and data elaborations which would be unfeasible otherwise.

The more this class of services becomes popular, the more the subscribers of these services need guarantees. In fact, the other side of the coin of cloud storage and computing applications is security: what happens to our data? Can we rely on our service providers? And are we able to conceal private information, without harming the performances of the service?

The straightforward approach to cloud services, where a client simply provides an online server with the data that must be stored and/or elaborated, relies on the assumptions that the service provider is *trusted* in two different senses. On the one hand, the user must be confident that the provider reliably stores his/her data, and carries out the required computations correctly. Unintentional errors or malicious behaviors (due, for instance, to economical reasons) could in fact affect the integrity of the data stored and/or elaborated at the server side, often in a way hardly detectable by the client. On the other hand, all outsourced data would be revealed to the provider, while they often contain information that the owner would like to keep confidential. The main problem is that these two requirements are apparently conflicting: the natural solution to enforce data confidentiality is cryptography, which however makes much harder to efficiently verify the integrity of the outsourced data and the correctness of the computations.

In the remainder of the chapter, we will focus on cloud storage applications, and discuss the main security concerns they pose. The main issue is to allow any customer to check the integrity of the outsourced data, while keeping them secret to the provider. To preserve both the performances of the system and the benefits of the end user, this features must be provided with the client directly retaining little or no information at all. Some solutions have been recently proposed to address this problem [146–149], denoted in the literature as *Provable Data Possession* (PDP). However, PDP is not the only concern, as we will motivate in the remainder of the chapter. In particular, we will underline the importance of the performances of the storage media used to store clients' data at the server side. To show how to efficiently verify the compliance of such performances to the clauses stated in the Service Level Agreement will be the purpose of Chapters 5 and 6.

## 4.1 Cloud Storage Main Issue: Provable Data Possession

The security requirements related to outsourcing the storage of private data to the cloud have been extensively investigated in the literature. To remotely check the integrity of such data was soon identified as one of the main concerns, but the first proposed approaches [150, 151] were very inefficient, requiring the server to exponentiate the whole file to produce the integrity proof.

As a consequence, several researchers tried to design schemes not requiring access to the entire file. Golle et al. [152] introduced the concept of *enforcing data storage complexity*. Relying on the assumption that a server may behave dishonestly only to reduce the complexity of its tasks, they proposed efficient schemes able to verify that the server is storing a file of the same size of the originally remotely stored file, but not necessarily the original file itself. Schwarz et al. [153] proposed a scheme based on *algebraic signatures*. Unfortunately, the scheme is unfeasible if the stored file is large, because the amount of data that the server must access is linear in the total size of the remotely stored file. Finally, Juels and Kaliski [154] introduced the notion of *Proof of Retrievability (PoR)*, which leverages error-correcting codes to ensure the possession and retrievability of a previously remotely stored file. However, PoR schemes can only be applied to encrypted files, and can only handle a limited number of queries, fixed in advance [155].

In the seminal paper [146], Ateniese et al. formally introduced the notion of *Provable Data Possession (PDP)*, and proposed the first two schemes able to efficiently verify the integrity of a remotely stored file. Both schemes enforce *probabilistic* PDP, that is, to only check the integrity of a few randomly chosen blocks, instead of the whole file. Regardless of which and how many blocks are challenged, thanks to so-called *homomorphic verifiable tags*, the client is only required to store a constant amount of data to verify the proof. Both schemes are provably-secure, and provide support for *public verifiability*, that is, to enable any authorized third party to perform the verification.

## 4.2 Extending Provable Data Possession

The work [146] somehow moved the target for all researchers in the area, that were not anymore asked to provide a brand new solution for PDP, but rather to extend or modify the idea used in [146].

The first concern became to improve the efficiency, both in terms of communication and computation overhead. Along this line, Ateniese et al. [147] proposed a light-weight PDP scheme, based on cryptographic hash functions, that also enables dynamic data operations. In [156], the authors proposed a solution whose maximum running time can be chosen at set-up time and traded off against client storage.

To deal with *dynamic data operations* (in opposition to append-only ones), in [149], based on a variant of *authenticated dictionaries* [157], rank information is used to organize dictionary entries. Zhu et al. used *hash index hierarchies* in [158], and simple *index hash tables* in [159]. Yang et al. [160] proposed a PDP scheme based on *Merkle Hash Trees (MHTs)* (already used for dynamically changing data in [161]) and on *bilinear signatures*, which minimizes both communication and computation overhead, and is suited for resource constrained mobile devices.

Some researchers investigated the possibility to enable *privacy preserving third party verification* [162, 163]. Hao et al. [162] used RSA-based homomorphic verifiable tags, while Wang et al. [163] proposed *randomly masked homomorphic verifiable tags*. Both schemes use previous solutions ([151] and [161], respectively) as a building block, inheriting from them the support for dynamic data operations.

Finally, among the advanced features that a PDP can support, *resilience to data loss or corruption* is for sure one of the priorities. To mitigate small file alterations, Ateniese et al. [155] proposed to use *forward error correction (FEC)* on the outsourced files, while Chen et al. [164] proposed to encode the outsourced data using *Reed Solomon (RS) codes* based on *Cauchy matrices* [165]. To protect against *Byzantine failures* (crashes, failing to receive or send requests, etc.) [166], several researchers proposed PDP schemes relying on *multiple replicas* [145, 148, 167].

Curtmola et al. [148] modified the generating process of the *homomorphic verifiable tags* proposed in [146], so that a single set of tags can be used to verify any number $n$ of replicas, being resilient to $n - 1$ failing servers. More generally, Wang et al. [145] proposed to first apply an $(m, k)$ RS-code, and then compute the homomorphic tags on the encoded file, which is distributed over $n = m + k$ different servers. This way, resilience to at most $k$ failures is provided.

In the context of multiple replica support, the protocol proposed by Bowers et al. [167] deserves a particular mention. Assuming that the Service Level Agreement (SLA) stipulated by the client and the service provider establishes that the provider will store several copies of the file on different disks, the solution described in [167] allows the client to verify that his/her outsourced data are indeed replicated on multiple disks. To the best of our knowledge, [167] is the only paper in the area not focusing on improved PDP solutions, but rather trying to emphasize the concept that *data possession* and *integrity* are not the only requirements that a client might need to *prove*.

Summing up, the PDP problem was addressed thoroughly in the literature, and many important features were added to try to make future cloud storage services as fulfilling as possible for the user. Nevertheless, there are many aspects of the problem that still deserve attention from the research community. In particular, (almost) all the aforementioned works focused on verifying whether the server is actually storing (somewhere/somehow) the remotely stored data. We argue that, beyond the data possession itself, a very important problem is the actual storage media used to keep the data at the server side. In fact, in most cases where data storage is outsourced is also very important that the data are actually accessible in a “short" time (this being defined in the SLA between the client and the server). This problem was raised in [168], showing that such a malicious behavior at the server side is indeed practicable. Unfortunately, so far there was still no solution in the literature that addressed the problem of detecting this malicious behavior in an efficient way. The purpose of Chapters 5 and 6 will be exactly to fill this gap.

## 4.3 PDP Protocol Description

This section is dedicated to the description of the PDP protocol originally proposed in [147]. The protocol is composed of two phases. The *setup phase* (presented in 4.3.1) consists of preliminary operations performed only once, concurrently to the outsourcing of the data to the server. The *challenge/response phase* (presented in 4.3.2), is the part of the protocol where the correct behavior of the service provider is actually verified. To describe this protocol in detail is particularly important, because the solution described in Chapters 5 and 6 is built upon this scheme. However, here we will only focus on how $\mathcal{U}$ can generate the challenge and check whether the response received is correct, hence obtaining PDP.

### 4.3.1 Setup Phase

During the setup phase, $\mathcal{U}$ splits the file $F$ into $n$ blocks $F[1], \ldots, F[n]$. It has to decide the maximum number $m$ of challenges it wants to be able to perform and the number $r$ of blocks to include in each challenge. $\mathcal{U}$ then has to randomly generate each challenge and the corresponding verification tokens.

Let $f$ and $g$ be two keyed pseudorandom functions with

$$\begin{aligned} f: &\quad \{0,1\}^{\alpha} \times \{0,1\}^{\mu} \to \{0,1\}^{\beta} \,, \\ g: &\quad \{0,1\}^{\beta} \times \{0,1\}^{\nu} \to \{0,1\}^{\nu} \,, \end{aligned}$$

where $\mu = \log_2 m$ and $\nu = \log_2 n$. In words, $f$ takes as inputs the index $i \in \{1, \ldots, m\}$ corresponding to one of the challenges and a $\alpha$-bits key ($h$) and returns a $\beta$-bits pseudorandom key ($k$); $g$ is a pseudorandom permutation of the set of the indexes $\{1, \ldots, n\}$, depending on a $\beta$-bits key ($k$). To distinguish between the keys and the real inputs to the functions, we will write $f_h(i)$ to denote $f(h,i)$ and $g_k(j)$ to denote $g(k,j)$. $\mathcal{U}$ has three $\alpha$-bits keys: $W, Z$ and $K$. For each challenge $C_i$, $\mathcal{U}$ first uses $W$ to generate the key $k_i = f_W(i)$ used to obtain the pseudorandom permutation $g_{k_i}$, which determines the indexes of the blocks included in $C_i$. The pseudorandom permutation $g_{k_i}$ is simply used to permit a smaller challenge, which includes only the session key $k_i$ and not all the blocks indexes. Then, $\mathcal{U}$ uses $Z$ to generate a pseudorandom value $c_i = f_Z(i)$ included in the challenge to prevent $\mathcal{P}$ from precomputing all possible tokens. The token corresponding to the challenge $C_i$ is computed as

$$v_i = H(c_i, F[g_{k_i}(1)], \ldots, F[g_{k_i}(r)]) \,,$$

where $H$ is a cryptographic hash function.

Finally, $K$ is used to encrypt the token with a symmetric encryption scheme $E_K$, obtaining $v_i' = E_K(i, v_i)$. The set $V$ of all tokens is eventually sent to $\mathcal{P}$ together with the file $F$. The setup phase is shown in Algorithm 1.

### 4.3.2 Challenge/Response Phase

Once the setup phase has been performed, the challenge/response phase can be carried out in a rather straightforward way.

Assume $i - 1 < m$ iterations of the protocol have already been performed. To generate the challenge $C_i$, $\mathcal{U}$ computes the key $k_i$, necessary for $\mathcal{P}$ to select the correct blocks of $F$, and the nonce $c_i$ that $\mathcal{P}$ has to include in the computation: the challenge is $C_i = (k_i, c_i)$. $\mathcal{P}$ can now

**Algorithm 1**: $\mathcal{U}$'s Setup Phase

```
Data: File $F = F[1], \dots, F[n]$
Result: Data to be stored by $\mathcal{P}$
begin
    Choose the number $m$ of challenges;
    Set $\nu = \log_2 n$ and $\mu = \log_2 m$;
    Choose security parameters $\alpha$,$\beta$ and functions $f, g$;
    Choose the number $r$ of indexes per challenge;
    Generate randomly the master keys $W, Z, K \in \{0,1\}^{\alpha}$;
    for $i \leftarrow 1$ to $m$ do
        Generate $k_i = f_W(i)$ and $c_i = f_Z(i)$;
        Compute $v_i = H(c_i, F[g_{k_i}(1)], \dots, F[g_{k_i}(r)])$;
        Compute $v_i' = E_K(i, v_i)$;
    end
    Set $V = \{(i, v_i') : 1 \leq i \leq t\}$;
    Send to $\mathcal{P}$: $(F, V, g)$;
end
```

compute the token $z = H(c_i, F[g_{k_i}(1)], \dots, F[g_{k_i}(r)])$, take the corresponding encrypted token $v_i'$ and send the response $R_i = (z, v_i')$ back to $\mathcal{U}$. Let $D_K = E_K^{-1}$ denote the decryption function corresponding to $E_K$. To verify that $R_i$ is valid, $\mathcal{U}$ decrypts $v_i'$, computing $D_K(v_i') = (i, v_i)$, and checks if $(i, v_i) = (i, z)$. The challenge/response and verification phase is shown in Algorithm 2.

**Algorithm 2**: Challenge/Response Phase

```
begin Protocol iteration $i$
    begin Challenge
        $\mathcal{U}$ computes $k_i = f_W(i)$ and $c_i = f_Z(i)$;
        $\mathcal{U}$ sends to $\mathcal{P}$: $C_i = (k_i, c_i)$;
    end
    begin Response
        $\mathcal{P}$ computes $z = H(c_i, F[g_{k_i}(1)], \dots, F[g_{k_i}(r)])$;
        $\mathcal{P}$ sends to $\mathcal{U}$: $R_i = (z, v_i')$;
    end
    begin Verification
        $\mathcal{U}$ computes $(i, v_i) = D_K(v_i')$;
        if $(i, v_i) = (i, z)$ then
            Verification passed successfully;
        else
            Verification failed;
        end
    end
end
```

We underline once again that the PDP protocol we just described is basically the same proposed in [147], where the authors present a complete analysis of the security of the scheme.

## 4.4 Summary

In this chapter, we provided an overview of the security and privacy issues related to Cloud Storage (CS) applications. In particular, we identified Provable Data Possession (PDP) as the main concern for end users, who need to be able to verify the integrity of the remotely stored data, without revealing the content of such data to the service provider, and directly retaining little or no information at all. Secure and practical solutions to address PDP have been recently proposed in the literature. In particular, we detailedly described the scheme proposed in [147], which is extremely computationally efficient on the server side, and therefore allows to generate integrity proofs very rapidly. The research community realized that PDP itself is not the only concern in CS systems: users can be particularly subsidized to rely on remote storage services if such services allow to perform dynamic data operations, to enable privacy preserving third party verification, or to recover from data loss or corruption. However, there are still many aspects of CS that need to be investigated. In particular, we stressed that to be only able to verify that the remotely stored data are correctly stored (somewhere, somehow) may not be enough in many application settings, and that data access time may be a critical aspect for many users to rely on CS. In the following chapters, we will therefore develop the concept of Provable Storage Medium (PSM), which responds to the idea that the minimum performances of the storage media used on the server side might be a fundamental part of the Service Level Agreement (SLA), which any user should be able to verify.

# 5 Introducing Provable Storage Medium

Delays in the retrieval time of remotely stored data can cause economic losses to the data owners (*e.g.*, when the remotely stored data are the content of an e-commerce website). It is therefore natural to assume that the Service Level Agreement, stipulated by clients and providers of any storage service, includes a clause stating that data will be stored in RAM. In this chapter, we define the concept of *Provable Storage Medium* (PSM), to denote the ability to verify the storage medium used by the provider. We further show how PSM strongly depends on a careful analysis of the system model, and on an accurate evaluation of all parameters and variables involved. The thorough analysis provided in this chapter will be the stepping stone for the scheme introduced in Chapter 6.

## 5.1 Introduction

As discussed in Chapter 4, the first concern of any costumer of a cloud storage service is to be able to check the integrity of the outsourced data. A straightforward approach would be to let the client downloads a full copy of the data stored at the server side, and compare them with a copy of the same data stored locally. A similar solution would however sacrifice one of the main features of cloud storage: the client being able to directly retain little or no information at all. The problem of making integrity checks both secure and efficient is usually denoted in the literature as Provable Data Possession (PDP). Many solutions have been recently proposed to address PDP [146–149], and are comprehensively surveyed in Section 4.1.

More generally, as recently observed in the literature [168], while complying to the data possession, a cloud provider might not respect other aspects of the Service Level Agreement (SLA). In particular, a key feature of these outsourced services is usually that the data must be available online (*e.g.*, being accessed through the Web) to the *data owner*, or in general to authorized users. In many situations, data access time is almost as important as data integrity. For instance, think to financial data needed to drive real-time share trading, or to retailer websites, whose customers are not willing to wait for the requested data more than 2–4 seconds [169, 170]. Delay introduced by violating the agreement could result in a direct financial loss in the former scenario, or in loosing possible customers in the latter one—that is, a financial loss as well—hence the compensatory measures introduced by a SLA.

However, it could happen that data retrieval takes more time than expected, because of circumstances independent from the behavior of the provider (*e.g.*, network overload). What can be included in the SLA are the technical measures that the provider commits to adopt, in order to fulfill the desired goals. In particular, what mainly affects data retrieval time is the mass storage used: different storage media exhibit very different access times. Consequently, we assume that the SLA states that the service provider faithfully maintains two copies of the remotely stored data, one on disk (to enhance reliability) and the other one in RAM (to ensure fast access), and that it precisely specifies the characteristics of the hardware used by the provider.

Similarly to PDP, the problem is how to enable the client to efficiently verify, ideally with any portable, resource-constrained device, that the provider is adhering to the SLA. We refer to this ability as *Provable Storage Medium* (PSM). Observe that the storage provider could be motivated by economic reasons to store data on some second level storage—indeed, disks and tapes are much cheaper than RAM. This could be applied in particular to the portion of data that is requested more rarely (*e.g.*, information for non popular items in online shops). The solution we propose is to embed an efficient and secure PDP scheme [147] with a check on the response time. We will show that the client can rightfully expect, and consequently require, that the response to a PDP challenge is delivered within a predictable time limit. Even if the unfaithful provider could adopt smart strategies to reduce the chances that its misbehaviors are detected, the underlying PDP scheme prevents it to simply delete a portion of the data, because data integrity violation would be (probabilistically) detected.

To the best of our knowledge, the work in the literature most similar to ours is the protocol introduced by Bowers et al. in [167]. Assuming that the SLA establishes that the provider will store several copies of the file on different disks, the authors deal with the problem of enabling the client to remotely and efficiently verify that the provider is actually adhering to that clause of the SLA. Similarly to our scheme, the protocol relies on time measurements for the time necessary to locate, retrieve, and deliver a random block stored on a drive. Since using multiple disks the server can perform many tasks in parallel, from the time required to return a response to a challenge, the client can infer the number of disk drives used by the service provider. However, not only the purpose of the two papers is completely different, but there are several reasons why our work is sensibly superior to [167]: (i) the authors consider a weaker attacker model, (ii) they require the client to store a local copy of the file, and, most importantly, (iii) they do not provide any theoretical framework for setting the decision threshold in general, but only explain experimentally how to set the threshold in some specific cases.

## 5.2 Scenario and Protocol Outline

In this section we describe the basic model of our system, namely its main functions, the main actors, their roles, and their interactions. We also delineate the threat model and sketch the outline of the proposed protocol, which will be discussed in detail in Section 4.3

### 5.2.1 Scenario

In our system, depicted in Fig. 5.1, a *Service Provider* ($\mathcal{P}$) offers a remote storage service to a *User* ($\mathcal{U}$). Potentially, multiple users could concurrently use the storage facilities of $\mathcal{P}$, but for the extents of our analysis only the interactions between a single $\mathcal{U}$ and $\mathcal{P}$ are relevant. The service is regulated by a set of clauses listed in a SLA. The most relevant clause states that $\mathcal{P}$ must store the files of $\mathcal{U}$, and must have them ready in RAM at any time. $\mathcal{U}$'s aim is to increase reliability, and to make data quickly available on-line, leveraging the sensible performance gap between RAM and HDD in terms of data retrieval time. $\mathcal{U}$ does not have direct access neither to the RAM nor to the HDDs of $\mathcal{P}$. However, $\mathcal{U}$ and $\mathcal{P}$ can communicate at any time through a network channel, whose characteristics play an important role in the design of our scheme. In particular, we assume that the average network delay and its variance can be determined by users prior to contracting the service, and included in the SLA.

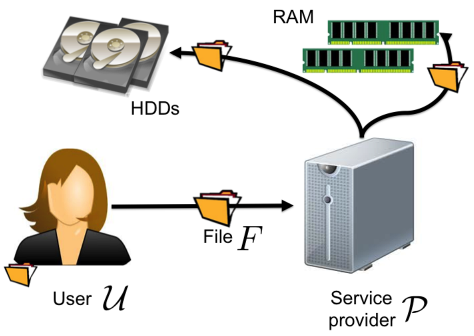


FIGURE 5.1: System Model: $\mathcal{U}$ sends a file $F$ to $\mathcal{P}$ and the latter stores it in its HDDs and uploads it into RAM.

### 5.2.2 Threat model

The service provider $\mathcal{P}$ is not trusted, indeed $\mathcal{P}$ is exactly the attacker $\mathcal{U}$ wants to defend from. $\mathcal{P}$ could misbehave and not adhere to the SLA signed with the users for several reasons, and in several ways. However, assuming that $\mathcal{P}$'s main purpose is to reduce the costs and increase its revenue, we observe two main attacks that $\mathcal{P}$ can mount to try to fool $\mathcal{U}$:

1. *Erasure attack*: $\mathcal{P}$ deletes a portion $h \in [0, 1]$ of the file and stores only the remaining fraction $1 - h$ in its storage facilities. This way, the provider could save some space and accept more clients without additional costs.
2. *Storage medium attack*: $\mathcal{P}$ stores a portion $\delta \in [0, 1]$ of the file only in the HDDs and the remaining fraction $1 - \delta$ also in RAM. This way, the provider can save money because RAM is more expensive than HDDs.

The parameters $h$ and $\delta$ describe the attacks level, that directly reflect their impact. To stress the importance of $\delta$ as a measure of the relevance of a storage medium attack, we introduce the concept of *$\delta$-malicious* provider, to denote a provider $\mathcal{P}$ which mounts a storage medium attack of parameter $\delta$. $\mathcal{P}$ is honest if $\delta = 0$, and totally dishonest if $\delta = 1$.

### 5.2.3 Goals and Desiderata

Our goal is to enforce what in the Introduction we defined as Provable Data Possession (PDP) and Provable Storage Medium (PSM). With the newly introduced notation, PDP can be defined as the ability to detect an erasure attack, so as to verify the integrity of the copy of the file $F$ stored by $\mathcal{P}$. Similarly, PSM can be defined as the ability to detect a storage medium attack, that is, to determine whether $\mathcal{P}$ is storing a copy of $F$ in RAM, or $F$ is only kept on disk and loaded on demand.

We want to achieve the aforementioned goals, without requiring $\mathcal{U}$ to locally store a copy of $F$. $\mathcal{U}$ only needs to store a limited amount of data, whose size is proportional to the number of allowed challenges.

### 5.2.4 Protocol Outline

To enforce both PDP and PSM, we propose to enrich a challenge/response based PDP scheme with response-time measurement. Even if the main idea can righteously be expected to apply to a wide range of possible PDP solutions, to permit a more precise theoretical and experimental analysis we focused on a specific PDP implementation. The PDP scheme we considered is *de* *facto* the same defined in [147], and it is described in details in Section 4.3. In short, the idea is to divide the stored file $F$ into $n$ blocks $F[1], \dots, F[n]$, and to let $\mathcal{U}$ challenge $\mathcal{P}$ by indicating a random selection of $r$ blocks. The required response is the hash of the concatenation of such blocks. Correct responses for all allowed challenges are precomputed during a *setup phase*, so that $\mathcal{U}$ can verify the correctness of the response without storing, nor entirely retrieving, a copy of the blocks involved. Since the block size is an important parameter in our construction, we specify that each block consists of 512KB of data, exactly as done in [147]. Time measurements performed on the user side cannot affect the security of the underlying PDP scheme, therefore the resilience to erasure attacks of our protocol is inherited by [147].

To implement PSM, we propose a very straightforward solution: $\mathcal{P}$ measures the time $T$ elapsed between the challenge transmission and the response reception, and check whether $T$ exceeds a predetermined threshold $\tau$. Similarly to the underlying PDP scheme, the protocol is probabilistic, in the sense that a single iteration only involves a fraction $r/n$ of the file $F$, and only verifies the correct storage of *the particular selected blocks*. However, there is a fundamental difference between PSM and PDP. If the challenge involves a deleted block, to escape detection $\mathcal{P}$ can only try to make an educated guess of the content of that block. Consequently, the success ratio of a PDP scheme only depends on the parameters $n$, $h$ and $r$. On the contrary, a PSM scheme must consider other technical aspects, in particular hardware specifications and network latency.

## 5.3 RAM and HDD Concepts

Our analysis is based on the different performances of RAM and HDD, in particular on the time necessary to retrieve $y$ randomly scattered 512KB blocks of data from these two storage media. In this section, we therefore accurately describe our RAM and HDD model.

### 5.3.1 RAM

In RAM, the minimum unit of data that can be read or written is called *block*;[1] while depending on architecture, it typically consists of 4KB. Any number of consecutive blocks can be accessed in a single operation: using the address bus, the processor sends to the RAM the physical address of the first block to access; the requested piece of data is accessed directly, independently of its location; finally, data are delivered to the processor through the data bus. The timing setting and the bus speed of the RAM memory module can be used to compute an upper-bound for the time needed to access a specific 4KB block, usually denoted as **access time** (**AT**), and whose order of magnitude is $10^{-8}$ seconds, that is, some tenths of nanoseconds (ns). The time necessary to

[1]Blocks of RAM and HDD must not be confused with the blocks of the scheme defined in 4.3: their sizes are very different (2-4KB vs. 512KB).

retrieve any number $m$ of consecutive 4KB blocks scales approximately linearly with $m$ as $m{\cdot}\mathrm{AT}$. Consequently, to retrieve a single 512KB block, that is, 128 consecutive 4KB blocks, it takes $t_{\mathtt{RAM}} = 128\mathrm{AT}$. Further, if the host needs to access $y$ randomly scattered data pieces from a RAM module, it needs to send $y$ independent requests. Thanks to direct access, independently on the location of the desired blocks, if all the pieces have the same size, each request takes the same time, and the total time required scales again linearly with $y$. In other words, the time needed to retrieve $y$ pieces of information, each one composed of 512KB (128 consecutive 4KB blocks), can be accurately approximated by $y \cdot t_{\mathtt{RAM}}$.

### 5.3.2 HDD

With respect to RAM, Hard Disk Drives (HDDs) work in a much more complicated way, where several factors may influence the time needed to recover a set of data blocks. HDDs are data storage devices with random access capabilities,[2] used to store and retrieve digital information using non-volatile memory. A HDD consists of several components, depicted in Fig. 5.2:

- **Platters**. Rotating disks covered by a coat of magnetic material from which data can be read and written. Platters rotate around a **spindle** that determines the rotation speed by using an electrical motor. Typically, rotation speed ranges from 4,200 to 15,000 revolutions per minute (rpm).
- **Heads**. Read/write devices mounted on moving **actuator arms** appended to an **actuator axis**. Heads are used to read/write magnetic information from/to the platters.

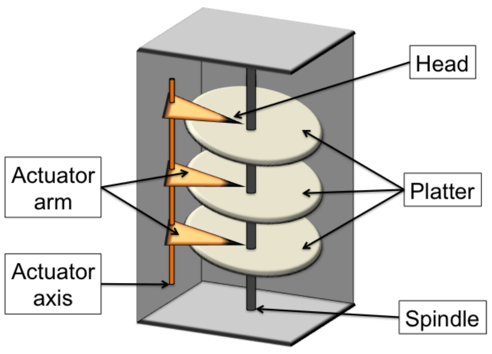


FIGURE 5.2: Main components of a hard disk drive.

Data stored in a HDD are typically organized into small chunks of 512B, called *sectors*. The minimum unit of data that a HDD can read or write is called *block*, and is usually composed of 4 sectors. Since information stored on platters is written and read by heads while platters

[2]In opposition to sequential access typical of tapes.

are spinning, sectors are stored along concentric circles called *tracks*. The *radial* speed of a platter is constant, but the *linear* speed gets clearly higher and higher from the innermost to the outermost border of the platter. If writing and reading frequencies were constant for the whole surface of the platter, tracks would all be composed of the same number of sectors, regardless of their length. Consequently, data density would not be the same on all tracks, with a remarkable loss of space efficiency due to data in outer tracks being sparser than data in inner tracks. For this reason, tracks of a platter in modern HDDs are divided into concentric *zones*: in each zone, tracks contain a different number of sectors, and data are correspondingly written and read at a different frequency. The number of zones, of tracks per zone, and of sectors per track for each zone may vary according to the size of the HDD and to design choices. However, a track is typically composed of 600 to 1200 sectors, corresponding to 300KB to 600KB of storage.

A good measure of the performance of a disk is the amount of data that it can send to the host computer in a second. This is generally indicated by the **Maximum Sustainable Transfer Rate** (**MaSTR**), measured as MB/s, that mainly depends on two factors (depicted in Fig. 5.3):

- **Average Seek Time** (**AST**): Measured in milliseconds (ms), is the average time it takes for the head to move to the track of the disk where data will be read or written.[3] The AST is computed as the weighted average of measured seek times of all possible seek combinations. Each seek time is measured from the start of the actuator's motion to the start of a read or write operation.
- **Rotation Speed** (**RS**): Measured in rpm, is the speed to which the platters of a HDD rotate. Based on the rotation speed, the **Revolution Time** (**RT**), that is, the time for a complete rotation measured in ms, can be easily computed. Similarly, the **Average Rotation Latency** (**ARL**), is computed by averaging over all possible rotation lengths. The ARL represents the average time it takes the head, once it is on the right track, to move to the sector where the data have to be read or written.

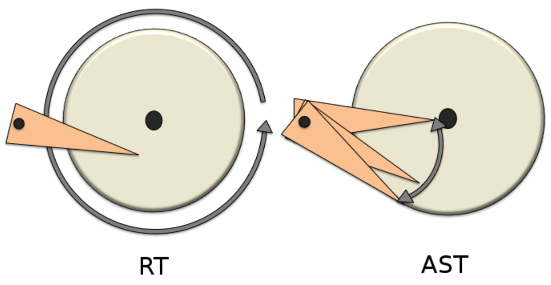


FIGURE 5.3: Graphical representation of the rotation and seek latencies.

To give an idea of the order of magnitude of the formerly introduced parameters for HDDs, in Table 5.1 we present real values taken from the datasheets of some common models of fast disks.

[3]We do not consider the controller overhead (which could add about 1ms to the seek time).

The current most important manufacturers of HDDs are Western Digital,[4] Seagate,[5] Toshiba,[6] Maxtor,[7] and Hitachi.[8] For each of the mentioned brands, we present the specifics of one or more models.

| **Model** | **AST** (ms) | **RS** (rpm) | **RT** (ms) | **ARL** (ms) | **MaSTR** (MB/s) |
|---|---|---|---|---|---|
| Seagate Cheetah 15K.7 SAS | 3.4 | 15,000 | 4.0 | 2.0 | 204 |
| Seagate Savvio 15K.3 SAS | 2.9 | 15,000 | 4.0 | 2.0 | 202 |
| Seagate Barracuda XT | n.a. | 7,200 | 8.33 | 4.16 | 149 |
| Maxtor Atlas 15K | 3.2 | 15,000 | 4.0 | 2.0 | 74.5 |
| Western Dig. WD RE | n.a. | 7,200 | 8.33 | 4.17 | 182 |
| Western Dig. WD XE | n.a. | 10,000 | 6.0 | 3.0 | 204 |
| Western Dig. VelociRaptor | n.a. | 10,000 | 6.0 | 3.0 | 200 |
| Toshiba AL13SEL | 3.7 | 10,500 | 5.7 | 2.8 | n.a. |
| Toshiba MBE2147RC | 2.9 | 15,000 | 4.0 | 2.0 | n.a. |
| Hitachi Ultrastar 15K300 SAS | 3.4 | 15,000 | 4.0 | 2.0 | 123-72 (zone 0-19) |

TABLE 5.1: Values for some HDDs on the market.

Summing up, the number of sectors per track in modern HDDs is variable, but when many (randomly selected) blocks are considered altogether, all tracks can be assumed to store exactly $S$ sectors, where $S$ is the average number of sectors per track. Assuming that the delay introduced by writing and reading operations is negligible with respect to the RT, the time to read a single sector is RT$/S$. Since storing a 512KB block of data requires $10^3$ sectors, we can denote the time needed to read a block of data as $t_R = 10^3\mathrm{RT}/S$.[9] To ease notation, let us also denote with $t_S$ the AST. We conservatively assume that the head is initially already on the track where the first block to retrieve is stored, but to retrieve a bunch of blocks, the head needs to *jump* from the end of a block to the beginning of the following one. Consequently, to retrieve $y$ randomly scattered blocks from a HDD, it takes $y \cdot t_R + (y-1) \cdot t_S$. Finally, we must consider the eventuality that $\mathcal{P}$ stores $N$ copies of the file $F$ on $N$ HDDs. This way, $\mathcal{P}$ can concurrently retrieve one block from *each one* of the $N$ disks in parallel in time $t_R$. If $\mathcal{P}$ must retrieve $y$ blocks in total, $\mathcal{P}$ must nevertheless sequentially retrieve $\lceil y/N \rceil$ blocks from the *same* disk. In other words, $N$ disks accessible in parallel allow $\mathcal{P}$ to retrieve $y$ blocks in time $\lceil y/N \rceil t_R + (\lceil y/N \rceil - 1)t_S$. Observe

[4]http://www.wdc.com/en/
[5]http://www.seagate.com/gb/en/
[6]http://www.toshiba.com
[7]http://www.maxtor.com
[8]http://www.hitachi.com
[9]We are assuming that a block is stored in consecutive sectors, that is the most favorable disposition for $\mathcal{P}$.

that all the former assumptions are *de facto* granting greater capabilities to a malicious provider, increasing its chances to produce a correct response sufficiently fast to avoid detection.

## 5.4 Hashing and Network Metrics

As we will see in Sections 6.1 and 6.2, the distribution of the hashing time, denoted $t_H$, and of the network delay, denoted $t_{\texttt{net}}$, can sensibly affect the reliability of the scheme. Unfortunately, $t_H$ and $t_{\texttt{net}}$ do not depend on design choices, nor they can be deduced by any hardware specification. Therefore, as part of the preliminary analysis presented in this chapter, we conducted extensive experiments to realistically model these two components.

### 5.4.1 Hashing Experiments

To get a realistic estimate for $t_H$, we studied the hash computation time over inputs of different size. In particular, we denoted $t_H(r)$ the time measured for the hashing of the concatenation of $r$ 512KB blocks, for $r$ varying in $\{1, 10, 100, 1000, 10000\}$ (corresponding to challenges involving up to 4GBs of data). For each $r$, we randomly created an input file containing exactly $r$ 512KB blocks, and repeated the hash computation 1000 times. The experiments were executed by a C-program, developed using the OpenSSL crypto library version (1.0.1c).[10] The results of the 1000 iterations for a specific $r$ were elaborated to compute the statistical mean $\mu_H(r)$ and standard deviation $\sigma_H(r)$. We report the results of the experiments in Table 5.2.

As expected, both the mean value $\mu_H(r)$ and the standard deviation $\sigma_H(r)$ grow approximately *linearly* with $r$. In particular, $\mu_H(r) \approx 2.2r$ ms and $\sigma_H(r) \approx 2.5r\,\mu$s represent two very accurate approximations, especially for large values of $r$. The occurrence of a positive variance can be explained by the fact that during each run of the hash computation the load of the CPU is different, resulting in a slightly different computation time. However, Table 5.2 underlines that, for each value of $r$, $\sigma_H(r)$ is three orders of magnitude smaller than $t_H(r)$. Therefore, we can conclude that $\sigma_H(r)$ is negligible.

| $r$ | $\mu_H(r)$ ($\mu$s) | $\sigma_H(r)$ ($\mu$s) |
|---:|---:|---:|
| 1 | 2226.811000 | 2.065739 |
| 10 | 22233.837000 | 26.620940 |
| 100 | 241835.252000 | 258.316431 |
| 1000 | 2219026.408000 | 2510.661525 |
| 10000 | 22189821.320000 | 25044.320500 |

TABLE 5.2: The results of the hashing experiment

[10]http://www.openssl.org/source/openssl-1.0.1c.tar.gz

### 5.4.2 Network Delay Experiments

To find a realistic estimate for $t_{\texttt{net}}$, we let four different clients,[11] located at the Amazon datacenters in Northern-Virginia, Sao-Paulo, Sydney, and Tokyo, measure the network access time to several web sites at different geographic locations. For this purpose, we chose four university domains: Cornell University (NY, USA, GMT-5), Kyoto University (Japan, GMT+9), University of Milan (Italy, GMT+1), and Berkeley University (CA, USA, GMT-8). To execute the experiment, we created at each one of our clients a `cron` job, running a script which calls the `fping` utility and directs the output to a data file. The `fping` utility measures the Round Trip Delay (RTD) of a Internet Control Message Protocol (ICMP) [171] echo packet directed to the four university websites. More precisely, over a whole week, on each day and on every hour, `cron` executed the script calling `fping`, which sent two hundred ICMP echo requests to the four university websites and measures their RTDs. To tune the packet payload size to simulate a message of the PSM protocol, the `fping` command was used with the "-b" option. The results of this experiment are illustrated in Figs. 5.4 and 5.5.

Fig. 5.4 shows the daily RTDs to Milan, averaged over the hours of the day. The measured values are pretty stable, with deviations always at least one order of magnitude smaller than the corresponding average: 15 ms vs. 100 ms from Northern-Virginia, 1 ms vs. 200 ms from Sao-Paulo, 10 ms vs. 287 ms from Tokyo, and 1 ms vs. 335 ms from Sydney.

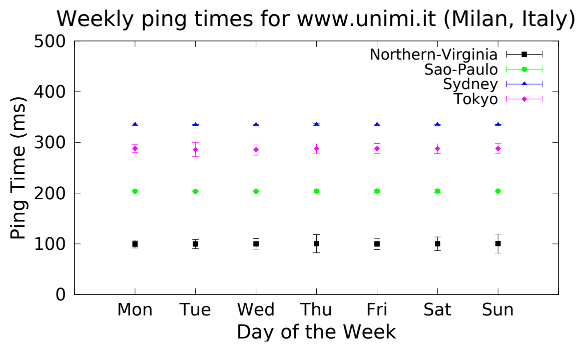


FIGURE 5.4: Daily average and standard deviation for the RTD to the website of the University of Milan

From Fig. 5.5, we can observe that the RTDs from all pinging locations (i.e., Northern-Virginia, Sao-Paulo, Sydney, and Tokyo) to all the university websites (i.e., www.cornell.edu, www.kyoto-u.ac.jp, www.unimi.it, and www.berkeley.edu) show a trend similar to what seen for Milan: the RTD time from each pinging location to each one of the university websites is stable around a specific value with a low standard deviation. These results indeed confirm the assumptions we made in Section 6.1, concerning the predictability of $t_{\texttt{net}}$ (i.e., RTD), and permit to accurately foresee its distribution.

[11]Precise specification of the clients used in the experiments, which are not needed for the comprehension of this section, will be given in Section 6.5.1.

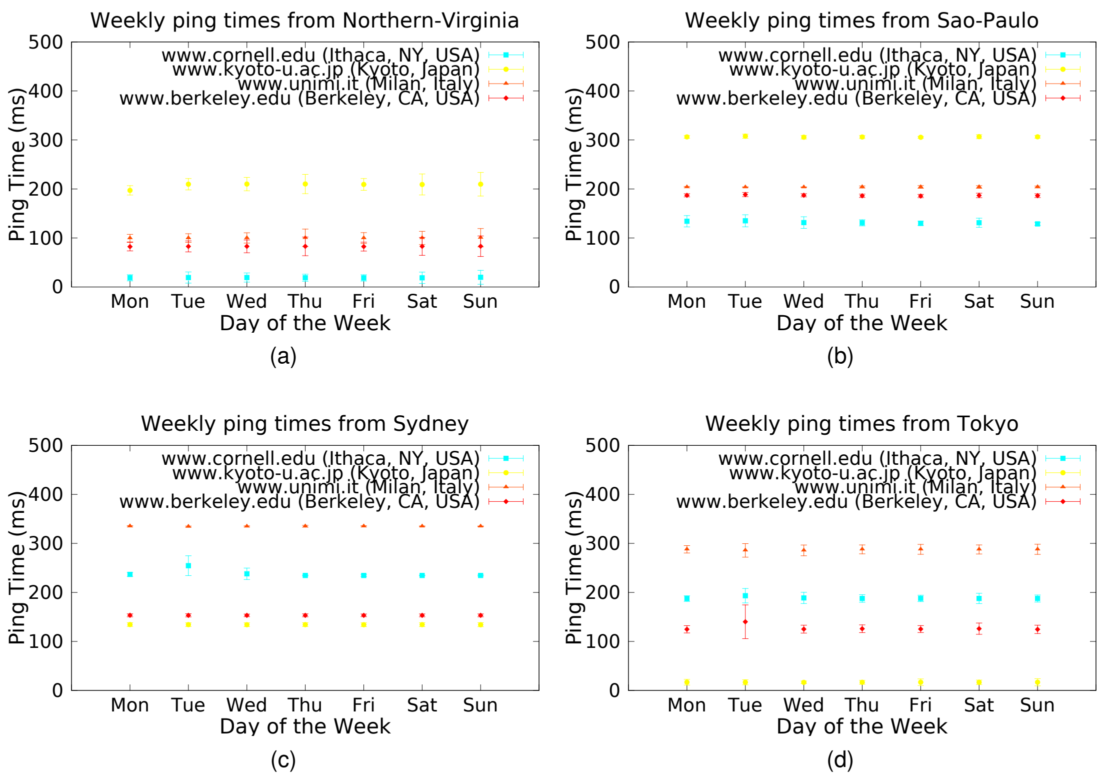


FIGURE 5.5: Daily average and standard deviation for the RTD, for all combinations of source and destination.

## 5.5 Summary

In this chapter, we introduced the concept of Provable Storage Medium (PSM), to describe the ability of a user of a Cloud Storage (CS) service to (probabilistically) verify that the performances of the storage media used by the provider comply with those stated in the Service Level Agreement (SLA). To address this novel problem represents a fundamental aspect of designing a reliable CS system, in particular as the pace of life is speeding upper and upper, and in many cases data must be retrievable in a very short time not to incur in dangers or economic losses. Contrarily to a proof of possession, whose success only depends on whether the remotely stored data are correctly stored or not, to provide PSM many variables need to be taken into account. The purpose of this chapter was exactly to delineate the system model, and in particular to understand: (i) what different strategies a dishonest provider can adopt, (ii) what are the typical performances of HDD and RAM modules, and (iii) if the network delay and the time needed to compute the proof of possession (in our case, hash functions) may conceal the delay introduced by a slower storage medium. The careful analysis provided in this chapter will be the starting point for the PSM scheme described in the next chapter.

# 6 Enforcing Provable Storage Medium

In this chapter, we finally show how probabilistic Provable Storage Medium (PSM) can be enforced, based on the rationale that network delay and RAM access are negligible, with respect to the time to retrieve data from disk (or other slower media). The proposed PSM protocol, built on a previous Provable Data Possession (PDP) solution, also checks for the integrity of the remotely stored data, requiring neither the local storage, nor the download from the provider, of such data. The thorough analysis of the system model presented in Chapter 5, together with a deep study of all the variables involved allow to tune the parameters of our scheme so as to obtain the desired false positive and false negative rates. An extensive simulation campaign confirms the quality and viability of our proposal.

## 6.1 Data Retrieval Time

In our scheme, to enforce PSM, $\mathcal{U}$ measures the time elapsed between the challenge transmission and the response reception. The total delay observed can be substantially decomposed into three main components:

$t_{\texttt{net}}$: Is the network delay time, sum of the time needed for the challenge to get to $\mathcal{P}$ and for the response to get back to $\mathcal{U}$.

$t_{\texttt{ret}}$: Is the information retrieval time, employed to retrieve the requested blocks $F[i_1], \ldots, F[i_r]$ from memory.

$t_H$: Is the hash computation time, required for $\mathcal{P}$ to compute the value $z = H(c_i, F[i_1], \dots, F[i_r])$ that must be included in the response.

$t_{\texttt{net}}$ and $t_H$ are not influenced by the storage medium used by $\mathcal{P}$ to store the requested blocks of the file $F$. $t_{\texttt{net}}$ only depends on external factors like network congestion. $t_H$ depends on the size $r$ of the challenge, and might vary according to the current load of the CPU while the hash is being computed[1]. On the contrary, the distribution of $t_{\texttt{ret}}$ *strongly* depends on $\mathcal{P}$'s behavior. Assuming that hardware specifications and challenge size $r$ are clear from the context, we denote $t_{\texttt{ret}}(\delta)$ the time needed by a $\delta$-malicious provider for the data retrieval operation.

The two extreme cases can be easily analyzed, based on the HDD and RAM model described in Section 5.3. We denoted $t_{\texttt{RAM}} = 128\text{AT}$ the time needed to access one of the 512KB blocks $F[i]$ of the remotely stored file $F$, where AT is the RAM access time for a 4KB data block. When $\mathcal{P}$ is honest, and stores the whole file $F$ in RAM, we have $\delta = 0$, and $t_{\texttt{ret}}(0) = r \cdot t_{\texttt{RAM}}$. Therefore, the total time necessary for an honest provider to get the challenge, compute the response and send it back to $\mathcal{U}$ is

$$T_{\texttt{hon}} = rt_{\texttt{RAM}} + t_H + t_{\texttt{net}}$$

When $\mathcal{P}$ is completely violating the SLA, not storing any block of $F$ in RAM, we have $\delta = 1$. If $N$ is the number of copies of $F$ stored by $\mathcal{P}$ on different HDDs, we know that the time necessary to retrieve the requested blocks is $t_{\texttt{ret}}(1) = \lceil r/N \rceil t_R + (\lceil r/N \rceil - 1)t_S$. Therefore, the total time necessary for a 1-malicious provider to get the challenge, compute the response and send it back to $\mathcal{U}$ is

$$T_{\texttt{mal}} = \lceil r/N \rceil t_R + (\lceil r/N \rceil - 1)t_S + t_H + t_{\texttt{net}}$$

The general case $\delta \in (0, 1)$ is more difficult to deal with. In fact, if $\delta \in (0, 1)$, the time

$$T_\delta = t_{\texttt{ret}}(\delta) + t_H + t_{\texttt{net}}$$

necessary for $\mathcal{P}$ to get the challenge, compute the response and send it back to $\mathcal{U}$, cannot be determined *a priori*, because, for a random challenge, it is impossible to know how many of the required blocks are only stored on disk. The following section discusses the distribution of $t_{\texttt{ret}}(\delta)$ more in details.

## 6.1.1 Distribution of $t_{\texttt{ret}}(\delta)$

Let us define $f_{\texttt{RAM}}(X) = (r - X)t_{\texttt{RAM}}$, $f_{\texttt{HDD}}(X) = (t_R + t_S)\lceil X/N \rceil - t_S$, $f(X) = \max\{f_{\texttt{RAM}}(X), f_{\texttt{HDD}}(X)\}$, and $g(X) = (t_R + t_S)X/N - t_S$ (see Fig. 6.1).

[1]For the sake of readability, and without loss of generality, we omit the dependency of $t_H$ on $r$, apart where strictly necessary. We basically use the notation $t_H(r)$ only in Section 5.4, where we empirically show that $t_H(r)$ grows approximately linearly with $r$.

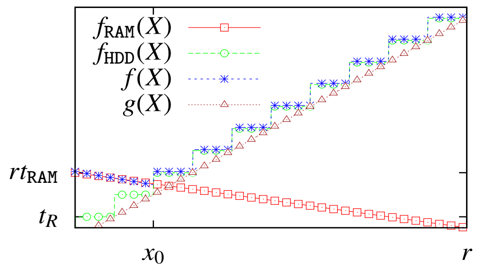


FIGURE 6.1: Graphical representation of $f_{\mathtt{RAM}}(X)$, $f_{\mathtt{HDD}}(X)$, $f(X)$ and $g(X)$.

$t_{\mathtt{ret}}(\delta)$ is precisely determined by the number $Y_D$ of blocks included in a challenge that are only stored on disk. In fact, if $\mathcal{P}$ works in parallel on $N$ disks storing a full copy of the file $F$, $\mathcal{P}$ must retrieve $r - Y_D$ blocks from RAM, and at least $\lceil Y_D/N \rceil$ blocks from one of the HDDs. In other words, $\mathcal{P}$ needs time $f_{\mathtt{RAM}}(Y_D)$ to retrieve data from RAM, and time $f_{\mathtt{HDD}}(Y_D)$ to retrieve data from disk, so that $t_{\mathtt{ret}}(\delta) = f(Y_D)$.

If $\delta \in (0, 1)$, $Y_D$ is a random variable[2], and so is $t_{\mathtt{ret}}(\delta)$, which is distributed as

$$\Pr[t_{\mathtt{ret}}(\delta) \leq z] = \Pr[f(Y_D) \leq z]$$

If we denote $x_0 = \min\{X : f_2(X) \geq f_1(X)\}$, we have $\Pr[f(Y_D) \leq z] = 0$ for all $z < (r - x_0)t_{\mathtt{RAM}}$.

More generally, the case $z \leq rt_{\mathtt{RAM}}$ is not very interesting, because $rt_{\mathtt{RAM}}$ is the time needed by an honest provider to retrieve the requested blocks. Even if $\mathcal{P}$ is not storing $F$ as agreed with $\mathcal{U}$, there is no reason to try (and it is impossible) to detect a malicious provider whose equipment allows to retrieve $r$ blocks faster than for an honest provider. Consequently, let us focus on the case $z > rt_{\mathtt{RAM}}$, which means

$$\Pr[t_{\mathtt{ret}}(\delta) \leq z] = \Pr[Y_D \leq f_{\mathtt{HDD}}^{-1}(z)]$$

Observe that $f_{\mathtt{HDD}}^{-1}(z)$ is a slight abuse of notation, because $f_{\mathtt{HDD}}(X)$, being a step function, is not invertible[3]. However, we observe that $f_{\mathtt{HDD}}(X) \geq g(X)$, so $g(Y_D)$ is a continuous lower bound for $t_{\mathtt{ret}}(\delta)$, and

$$\Pr[t_{\mathtt{ret}}(\delta) \leq z] \leq \Pr[g(Y_D) \leq z] \tag{6.1}$$

Even if we are sacrificing some precision, this bound permits to easily translate the dependency of $Y_D$ on $\delta$ and the other parameters into the dependency of $t_{\mathtt{ret}}(\delta)$ on the same variables.

[2]The distribution of $Y_D$ clearly depends on $\delta$, but here and later in the paper we omit such a dependence to ease notation.

[3]For every $z$, either $z$ has no preimages, or $z$ has multiple preimages: all the integer values in the interval $\left(\frac{z-t_R}{t_R+t_S}, \frac{z-t_R}{t_R+t_S} + 1\right]$.

### 6.1.2 Predicting the Value of $Y_D$

We just showed that $t_{\mathtt{ret}}(\delta)$ is completely determined by $Y_D$, that is, the random variable measuring the number of blocks of the challenge that $\mathcal{P}$ stores only on disk. In order to predict the value of $Y_D$, we can divide the $n$ blocks of $F$ into $\delta n$ *good* blocks, corresponding to blocks only stored on disk, and $(1 - \delta)n$ *bad* blocks, corresponding to those also stored in RAM. A random challenge $C$ can be described as a random extraction of $r$ blocks without replacement from such a population of $n$ blocks, and $Y_D$ is distributed as the number of good blocks picked. In other words, $Y_D$ is distributed as a Hypergeometric of parameters $(n, \delta n, r)$, and has mean $\mathbb{E}[Y_D] = \delta r$. The following theorem provides a Chernoff-like bound for the tail $Y_D \leq y$. To improve readability, we defer the proof to Section 6.3.1.

**Theorem 6.1.1.** *Assume that, for $\delta \in (0, 1)$, $\mathcal{P}$ stores $\delta n$ blocks of $\mathcal{U}$'s file $F$ only on disk. If $\mathcal{U}$ randomly chooses $r$ blocks to include in a challenge $C$, and if $Y_D$ denotes the number of such blocks which are only stored on disk, it holds, for all $y < \delta r$,*

$$\Pr[Y_D \leq y] \leq \exp\left[-\frac{2}{r}(\delta r - y)^2\right] \tag{6.2}$$

□

Observe that, based on Theorem 6.1.1, the minimum $r$ such that $\Pr[Y_D \leq y] \leq \epsilon$, for any negligible $\epsilon$, can be found imposing

$$\exp\left[-\frac{2}{r}(\delta r - y)^2\right] \leq \epsilon$$

which, under the condition $y < \delta r$, is satisfied for

$$r \geq \delta^{-1} y + \frac{\delta^{-2} \ln \epsilon^{-1}}{4}\left(1 + \sqrt{1 + \frac{8\delta y}{\ln \epsilon^{-1}}}\right) \tag{6.3}$$

## 6.2 Parameters Setting for PSM

Our protocol works on a challenge/response basis, where, if $\mathcal{U}$ receives a wrong response, it concludes that $\mathcal{P}$ deleted a portion of $F$. Reasonably, this is the most serious possible violation to the SLA. Therefore, when focusing on PSM, we can assume that $\mathcal{U}$ received a valid response to its challenge, since otherwise $\mathcal{U}$ would not be interested in the storage media actually used by $\mathcal{P}$.

To verify $\mathcal{P}$'s compliance to the SLA, $\mathcal{U}$ measures the time $T$ required to obtain a valid response. Based on $T$, $\mathcal{U}$ must be able to distinguish if the response was computed upon data only stored in RAM, or upon data at least partially stored on disk. The PSM scheme works as a statistical hypothesis test, relying on the assumption that an honest provider retrieves data faster than a malicious one: $\mathcal{U}$ sets a threshold $\tau$ and considers $\mathcal{P}$ honest if $T < \tau$, while malicious otherwise.

$\mathcal{U}$ sets the test up according to a threshold $\delta_{\max}$, which corresponds to the maximum degree to which $\mathcal{U}$ is willing to tolerate a violation to the SLA. This could seem counterintuitive, but it is indeed necessary, because the time needed by $\mathcal{P}$ to retrieve data varies continuously as a function of $\delta$, so to fix a threshold for the response time corresponds to fix a threshold for the choice of $\delta$ as well. In this section, we show how $\mathcal{U}$ can tune the challenge size $r$ and the threshold $\tau$ so as to be able to distinguish whether $\mathcal{P}$ is honest or malicious (indeed, $\delta$-malicious, for $\delta \geq \delta_{\max}$), with negligible false positive and false negative probabilities. As expectable, the success rate of our PSM scheme turns out to depend on the difference $\mathbb{E}[t_{\mathtt{ret}}(\delta_{\max})] - rt_{\mathtt{RAM}}$ between the expected retrieval time of a $\delta_{\max}$-malicious and an honest provider, respectively. In particular, $\mathcal{U}$ needs to ensure that such a distance is sufficiently large with respect to the variance of $t_H$ and $t_{\mathtt{net}}$. Luckily, experiments run in Section 5.4 will show that the latter two variables are indeed highly predictable.

### 6.2.1 Test Description

As already outlined, the scheme relies on a standard statistical hypothesis test: to decide which one of two alternative hypotheses $H_0$ and $H_1$ ($\mathcal{P}$ being honest vs. $\mathcal{P}$ being malicious) is correct, identify a variable ($T$) which behaves differently under $H_0$ and $H_1$, and check if the measured occurrence of such a variable is more compatible with $H_0$ or $H_1$. Let $T_{\mathtt{hon}}$ and $T_\delta$ be defined as in Section 6.1. Formally, the two statistical hypotheses are:

$H_0$: $\mathcal{P}$ is honest and the time needed to get back the response is distributed as $T_{\mathtt{hon}}$.

$H_1$: $\mathcal{P}$ is $\delta_{\max}$-malicious and the time needed to get back the response is distributed as $T_{\delta_{\max}}$.

To infer about $\mathcal{P}$'s behavior, $\mathcal{U}$ sets a threshold $\tau$ and considers $\mathcal{P}$ honest if it can return the response within time $\tau$, and malicious otherwise. The probability of a false positive, that is, to wrongly accuse an honest provider, is given by $p_{\mathtt{fp}} = \Pr[T_{\mathtt{hon}} \geq \tau]$. On the other hand, the probability of a false negative, that is, not to detect a malicious provider, is $p_{\mathtt{fn}} = \Pr[T_{\delta_{\max}} < \tau]$. Contrarily to the former, the latter strongly depends on $\delta_{\max}$: for any fixed $\tau$, to obtain the desired $p_{\mathtt{fn}}$, the smaller is $\delta_{\max}$, the larger must be the size $r$ of the challenge. Since all the possible challenges, and hence their size $r$, are defined at setup time, for any choice of $p_{\mathtt{fn}}$, only $\delta$-malicious providers with $\delta$ larger than $\delta_{\max}$ will be detected with probability at least $1 - p_{\mathtt{fn}}$. Therefore, we assume that maximum accepted values $p_{\mathtt{fp}\max}$, $p_{\mathtt{fn}\max}$, and $\delta_{\max}$ are set by $\mathcal{U}$ at setup time. Fig. 6.2 gives a graphical representation of the strategy we propose. Note that it is only a clarifying example, since the PDFs (probability density functions) of $T_{\mathtt{hon}}$ and $T_\delta$ were drawn arbitrarily.

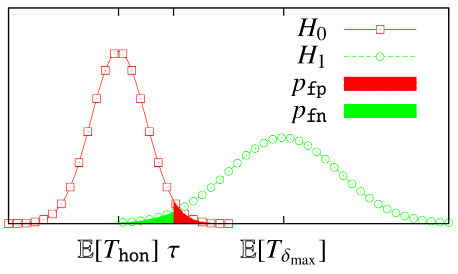


FIGURE 6.2: Graphical representation of the statistical hypothesis test.

Apparently, $p_{\mathtt{fp}}$ and $p_{\mathtt{fn}}$ cannot be both bounded at will, since, by decreasing $\tau$, $p_{\mathtt{fp}}$ decreases but $p_{\mathtt{fn}}$ increases, and vice versa. However, leveraging that $t_{\mathtt{RAM}}$ is typically 5 orders of magnitude smaller than $t_R$ and $t_S$ (tenths of ns vs. some ms), in Section 6.4 we will show that, by *realistically* increase $r$, it is possible to distance the two Gaussian bells corresponding to $H_0$ and $H_1$, until the two tails are very slightly overlapping. This way, $\tau$ can be set so as to make $p_{\mathtt{fp}}$ and $p_{\mathtt{fn}}$ negligibly small at the same time.

### 6.2.2 Tuning the Parameters of the Test

In Section 5.4, we will experimentally analyze the distribution of $t_H$ and $t_{\mathtt{net}}$. For the time being, let us simply assume that $t_H \overset{\mathrm{d}}{\sim} \mathcal{N}(\mu_H(r), \sigma^2_H(r))$ and $t_{\mathtt{net}} \overset{\mathrm{d}}{\sim} \mathcal{N}(\mu_{\mathtt{net}}, \sigma^2_{\mathtt{net}})$.[4] Now, let $\mu(r) = \mu_H(r) + \mu_{\mathtt{net}}$ and $\sigma^2(r) = \sigma^2_H(r) + \sigma^2_{\mathtt{net}}$, so that

$$t_H + t_{\mathtt{net}} \overset{\mathrm{d}}{\sim} \mathcal{N}(\mu(r), \sigma^2(r))$$

In the following, in all steps of the analysis where the dependency on $r$ is not relevant, for the sake of clarity we will denote $\mu = \mu(r)$ and $\sigma = \sigma(r)$.

The following theorem describes how the parameters of the statistical hypothesis test should be set, so as to obtain the desired false positive and false negative rates. Again, for the sake of readability, we defer the proof to Section 6.3.2.

**Theorem 6.2.1.** *Assume $\mathcal{U}$ implements the statistical hypothesis test described in Section 6.2.1, with hypotheses $H_0$ and $H_1$.*

[4] $\mathcal{N}(\mu, \sigma^2)$ denotes a Normal variable of mean $\mu$ and variance $\sigma^2$.

*(a) If $\tau$ denotes the threshold used for the test, the false positive and false negative probabilities satisfy*

$$p_{\mathtt{fp}} = 1 - \Phi\left(\frac{\tau - rt_{\mathtt{RAM}} - \mu(r)}{\sigma(r)}\right)$$

$$p_{\mathtt{fn}} \leq \frac{\rho}{\sqrt{4r\sigma^2(r) + \rho^2}} \cdot \exp\left[-\frac{2r\left((\delta_{\max}\rho - t_S + \mu(r)) - \tau\right)^2}{4r\sigma^2(r) + \rho^2}\right]$$

*where $\rho = r(t_R + t_S)/N$, and $\Phi$ is the CDF of the Standard Normal distribution.*

*(b) For any $p_{\mathtt{fp}\max}, p_{\mathtt{fn}\max} \in (0,1)$, both conditions*

$$p_{\mathtt{fp}} \leq p_{\mathtt{fp}\max} \qquad p_{\mathtt{fn}} \leq p_{\mathtt{fn}\max}$$

*hold concurrently, provided that $\mathcal{U}$ sets $r$ such that $\tau_{\max}(r) \geq \tau_{\min}(r)$, and $\tau \in [\tau_{\min}(r), \tau_{\max}(r)]$, where*

$$\tau_{\min}(r) = rt_{\mathtt{RAM}} + \sigma(r)\Phi^{-1}(1 - p_{\mathtt{fp}\max}) + \mu(r)$$

$$\tau_{\max}(r) = (\delta_{\max}\rho - t_S + \mu(r)) - \sqrt{\frac{4r\sigma^2(r) + \rho^2}{2r}\ln\left(\frac{\rho}{p_{\mathtt{fn}\max}\sqrt{4r\sigma^2(r) + \rho^2}}\right)}$$

□

The minimum acceptable value for $r$ is the one realizing $\tau_{\max}(r) = \tau_{\min}(r)$, in which case $\mathcal{U}$ has to set $\tau = \tau_{\min}(r) = \tau_{\max}(r)$. Otherwise, $\tau$ can be taken in the whole interval $[\tau_{\min}(r), \tau_{\max}(r)]$: a smaller $\tau$ means to prioritize $p_{\mathtt{fn}}$, while a larger $\tau$ means to favor $p_{\mathtt{fp}}$.

Since $\tau_{\max}(r)$ grows faster than $\tau_{\min}(r)$ (as we will verify in Section 6.5), there always exists an $r$ large enough to ensure $\tau_{\max}(r) \geq \tau_{\min}(r)$. However, it may happen that such an $r$ is too large for realistic applications. In that case, $\mathcal{U}$ has two choices. The first is to increase $\delta_{\max}$, which allows to decrease $r$ though maintaining the same false negative probability $p_{\mathtt{fn}}$. The second option is to make a choice between false positives and false negatives, that is, to choose if it is more important to identify a malicious provider or to avoid to wrongly accuse an honest one. If (reasonably) $\mathcal{U}$ decides to put false negatives first, $\mathcal{U}$ sets the threshold as $\tau = \tau_{\max}(r_{\max})$, where $r_{\max}$ is the maximum possible dimension of the challenge. The corresponding false positive probability will be given by

$$p_{\mathtt{fp}} = 1 - \Phi\left(\frac{\tau_{\max}(r_{\max}) - r_{\max}t_{\mathtt{RAM}} - \mu(r)}{\sigma(r)}\right)$$

### 6.2.3 Multiple Challenges

In the previous sections, we showed that the effectiveness of the scheme strongly depends on two parameters: the number $r$ of blocks of file included in a challenge, and the maximum degree $\delta_{\max}$ to which the user $\mathcal{U}$ is willing to tolerate a violation to the SLA. In some circumstances, to correctly infer on the honesty of the provider $\mathcal{P}$ with a single challenge/response iteration, $\mathcal{U}$ may need to excessively increase $r$ and/or $\delta_{\max}$. With the aim to provide a comprehensive analysis of our approach, in this section we describe an alternative strategy to improve the effectiveness of the scheme without increasing $r$ or $\delta_{\max}$. The approach consists in performing multiple challenges to the same provider, and draw a conclusion on its behavior based on the outcomes of the different iterations altogether.

Assume $\mathcal{U}$ performs $m$ iterations of the scheme and let $t_i, i = 1, \ldots, m$, denote the time elapsed from the transmission of the $i$-th challenge to the reception of the corresponding response. For each iteration, $\mathcal{U}$ checks if $t_i \leq \tau$ is satisfied. In this case, $\mathcal{U}$ must choose an integer $l \leq m$ and decide whether $\mathcal{P}$ is honest based on whether at least $l$ of the $t_i$'s satisfy $t_i \leq \tau$. The false negative probability, that is, the probability that at least $l$ out of the $m$ responses are delivered within time $\tau$ given statistical hypothesis $H_1$, becomes then

$$P_{\mathtt{fn}} = \sum_{k=l}^{m} \binom{m}{k} p_{\mathtt{fn}}^{k} (1 - p_{\mathtt{fn}})^{m-k}$$

where $p_{\mathtt{fn}}$ denotes the false negative probability of a single iteration. $P_{\mathtt{fn}} < \epsilon$, for any choice of $\epsilon$, can be obtained increasing $m$, or decreasing $l$ for $m$ fixed. The corresponding false positive probability is given by

$$P_{\mathtt{fp}} = \sum_{k=m-l+1}^{m} \binom{m}{k} p_{\mathtt{fp}}^{k} (1 - p_{\mathtt{fp}})^{m-k}$$

Observe that the challenges must be separated from each other by a sufficiently large time interval. This is to avoid that $\mathcal{P}$ retrieves the whole file $F$ when it gets the first challenge and keeps it in RAM until the last challenge arrives. In fact, while doing it only for a short time can be considered a way to cheat, if $\mathcal{P}$ behaves this way for challenges delivered very far away one from the other in time, then $\mathcal{P}$ is substantially behaving honestly.

## 6.3 Proofs

In this section, we provide the proofs to the main results of this chapter: Theorem 6.1.1 and Theorem 6.2.1

### 6.3.1 Proof of Theorem 6.1.1

First of all, let us recall a result by R. Impagliazzo and V. Kabanets [172], which is a generalization of the well known Chernoff Bound for dependent, but negatively correlated, variables. Application of Theorem B will be the essential step for the proof of Theorem 6.1.1.

**Theorem B (1.1 in [172]).** *Let* $X_i \in \{0, 1\}$*,* $i = 1, \dots, m$*, be boolean variables such that, for some* $0 \le p \le 1$*, and for every subset* $\Omega \subset \{1, \dots, m\}$*, it holds*

$$\Pr[\wedge_{i\in\Omega} X_i = 1] \le p^{|\Omega|}$$

*Then, for any* $p \le \gamma \le 1$*, it holds*

$$\Pr\left[\sum_{i=1}^{m} X_i \ge \gamma m\right] \le \exp\left[-mD_{KL}(\gamma\|p)\right] \le \exp\left[-2m(\gamma - p)^2\right] \tag{6.4}$$

□

$D_{KL}(\gamma\|p)$ denotes the *Kullback-Leibler Divergence*. For binary probability distributions it is defined as

$$D_{KL}(\gamma\|p) = \gamma \ln \frac{\gamma}{p} + (1 - \gamma) \ln \frac{1 - \gamma}{1 - p}$$

with $0 \ln 0$ interpreted as 0, since $\lim_{x\to 0} x \ln x = 0$. The condition $D_{KL}(\gamma\|p) \ge 2(\gamma - p)^2$, used in (6.4), follows easily from the convexity of $D_{KL}(\gamma\|p)$.

For $i = 1, \dots, r$, let $Y_i$ be the Bernoulli variable which indicates whether the $i$th block of the challenge is only stored on disk ($Y_i = 1$) or it is stored in RAM as well ($Y_i = 0$), so that $Y_D = \sum_{i=1}^{r} Y_i$. Since blocks are picked *without replacement*, the $Y_i$'s are not independent, reason why $Y_D$ is a Hypergeometric and not a Binomial.

Let $X_i = 1 - Y_i$, so that $\Pr[X_i = 1] = 1 - \delta$ is the ratio of blocks stored in RAM over total number of blocks. The $Y_i$'s being mutually dependent, the $X_i$'s are dependent too. In particular, if $i' \ne i$, $\Pr[X_{i'} = 1 | X_i = 1] < 1 - \delta$, since, once we pick a block stored in RAM, the ratio of blocks stored in RAM decreases. As a consequence, for every $\Omega \subset \{1, \dots, r\}$,

$$\Pr[\wedge_{i\in\Omega} X_i = 1] < (1 - \delta)^{|\Omega|}$$

and we can apply Theorem B with $\gamma = 1 - \delta + \alpha$, to obtain

$$\Pr\left[\sum_{i=1}^{r} X_i \ge (1 - \delta)r + \alpha r\right] \le \exp\left[-2r\alpha^2\right] \tag{6.5}$$

Now, if we observe that

$$\sum_{i=1}^{r} X_i = \sum_{i=1}^{r}(1 - Y_i) = r - \sum_{i=1}^{r} Y_i = r - Y_D$$

we see that the events

$$\sum_{i=1}^{r} X_i \geq (1 - \delta)r + \alpha r \qquad \text{and} \qquad Y_D \leq \delta r - \alpha r$$

are equivalent, and (6.5) can be restated as

$$\Pr[Y_D \leq \delta r - \alpha r] \leq \exp\left[-2r\alpha^2\right]$$

Finally, if we impose $y = \delta r - \alpha r$, we obtain

$$\Pr[Y_D \leq y] \leq \exp\left[-2r\left(\delta - \frac{y}{r}\right)^2\right]$$

for all $y < \delta r = \mathbb{E}[Y_D]$, which is equivalent to the desired bound

$$\Pr[Y_D \leq y] \leq \exp\left[-\frac{2}{r}(\delta r - y)^2\right]$$

by just explicating the dependency on the distance between $y$ and $\mathbb{E}[Y_D]$. □

### 6.3.2 Proof of Theorem 6.2.1

We will separately prove statements (a) and (b) of the theorem.

#### Proof of statement (a)

We start with the easier case, that is, computing $p_{\mathtt{fp}}$. We recall that

$$T_{\text{hon}} = rt_{\mathtt{RAM}} + t_H + t_{\mathtt{net}}$$

Consequently

$$p_{\mathtt{fp}} = \Pr[T_{\text{hon}} > \tau] = \Pr[t_H + t_{\mathtt{net}} > \tau - rt_{\mathtt{RAM}}] = 1 - \Pr[t_H + t_{\mathtt{net}} \leq \tau - rt_{\mathtt{RAM}}] = 1 - \Phi\left(\frac{\tau - rt_{\mathtt{RAM}} - \mu(r)}{\sigma(r)}\right)$$

Now, let $F_X(x)$ and $f_X(x)$ denote, respectively, the CDF and the PDF of a random variable $X$. To bound $p_{\mathtt{fn}}$, we need the following theorem.

**Theorem C .** *Let $U \overset{\mathrm{d}}{\sim} \mathcal{N}(\mu_U, \sigma_U^2)$ be normally distributed with mean $\mu_U$ and variance $\sigma_U^2$. Let $V$ be a random variable which satisfies, for all $v < \mathbb{E}[V]$, and for suitable parameters $a$ and $A$,*

$$F_V(v) \leq \exp\left[-a(A-v)^2\right]$$

*If $W = U + V$, then the CDF of $W$ satisfies*

$$F_W(w) \leq \frac{1}{\sqrt{2a\sigma_U^2+1}} \exp\left[-\frac{a((A+\mu_U)-w)^2}{2a\sigma_U^2+1}\right] \tag{6.6}$$

*Proof.* It is well known that

$$f_U(u) = \frac{1}{\sigma_U\sqrt{2\pi}} \exp\left[-\frac{(u-\mu_U)^2}{2\sigma_U^2}\right]$$

and that, if $W = U + V$, it holds

$$F_W(w) = \int_{-\infty}^{+\infty} F_V(w-u) f_U(u) du$$

By hypothesis, then, we have

$$F_W(w) \leq \frac{1}{\sigma_U\sqrt{2\pi}} \int_{-\infty}^{+\infty} \exp\left[-a(A-(w-u))^2 - \frac{(u-\mu_U)^2}{2\sigma_U^2}\right] du \tag{6.7}$$

For the sake of simplicity, let us define

$$\alpha = a + \frac{1}{2\sigma_U^2} = \frac{2a\sigma_U^2+1}{2\sigma_U^2} \qquad \beta = \frac{a}{a+\frac{1}{2\sigma_U^2}} = \frac{2a\sigma_U^2}{2a\sigma_U^2+1}$$

$$\gamma = \frac{\frac{1}{2\sigma_U^2}}{a+\frac{1}{2\sigma_U^2}} = \frac{1}{2a\sigma_U^2+1} \qquad \phi = \frac{a \cdot \frac{1}{2\sigma_U^2}}{a+\frac{1}{2\sigma_U^2}} = \frac{a}{2a\sigma_U^2+1}$$

After some computations, (6.7) can be restated as

$$\begin{aligned} F_W(w) \leq & \frac{1}{\sigma_U\sqrt{2\pi}} \int_{-\infty}^{+\infty} \exp\left[-\phi((A+\mu_U)-w)^2 - \alpha(u-(\beta(A-w)+\gamma\mu_U))^2\right] du \\ = & \frac{1}{\sigma_U\sqrt{2\alpha}} \exp\left[-\phi((A+\mu_U)-w)^2\right] \cdot \int_{-\infty}^{+\infty} \sqrt{\frac{\alpha}{\pi}} \exp\left[-\alpha(u-(\beta(A-w)+\gamma\mu_U))^2\right] du \end{aligned}$$

The expression in the integral is the density of a Normal variable $W$ with mean $\mu_W = \beta(A-w) + \gamma\mu_U$ and variance $\sigma_W^2 = 1/2\alpha$, so the integral evaluates to 1. Finally, we have

$$F_W(w) \leq \frac{1}{\sigma_U\sqrt{2\alpha}} \exp\left[-\phi((A+\mu_U)-w)^2\right]$$

and, recalling the definition of $\alpha$ and $\phi$, we obtain the desired result

$$F_W(w) \leq \frac{1}{\sqrt{2a\sigma_U^2 + 1}} \exp\left[-\frac{a((A+\mu_U) - w)^2}{2a\sigma_U^2 + 1}\right]$$

$\square$

In Section 6.1.1, we defined

$$g(X) = \frac{t_R + t_S}{N} X - t_S$$

and showed that $t_{\texttt{ret}}(\delta) \geq g(Y_D)$. Consequently, if we define

$$T'_\delta = g(Y_D) + t_H + t_{\texttt{net}}$$

we know that

$$T_\delta = t_{\texttt{ret}}(\delta) + t_H + t_{\texttt{net}} \geq T'_\delta$$

and that

$$\Pr[T_\delta \leq \tau] \leq \Pr[T'_\delta \leq \tau]$$

From (6.2), we know that

$$\begin{aligned}\Pr[g(Y_D) \leq z] &= \Pr[Y_D \leq g^{-1}(z)] \leq \exp\left[-\frac{2}{r}\left(\delta r - g^{-1}(z)\right)^2\right] \\ &= \exp\left[-\frac{2N^2}{r(t_R+t_S)^2} \cdot \left(\left(\frac{\delta r(t_R+t_S)}{N} - t_S\right) - z\right)^2\right]\end{aligned}$$

For the sake of clarity, let us define

$$\rho = \rho(r, t_R, t_S, N) = \frac{r(t_R+t_S)}{N} = (t_R + t_S) \cdot \frac{r}{N}$$

$\rho$ can be seen as the product of the time to retrieve a block from a disk ($t_R + t_S$), times the average number of blocks of challenge per disk ($r/N$), that is, as the average time necessary to recover all the blocks of a challenge. Now, we can restate the CDF of $g(Y_D)$ as

$$\Pr[g(Y_D) \leq z] \leq \exp\left[-\frac{2r}{\rho^2}\left((\delta\rho - t_S) - z\right)^2\right]$$

Recalling that $t_H + t_{\texttt{net}} \overset{\mathrm{d}}{\sim} \mathcal{N}(\mu, \sigma^2)$, we can apply Theorem C with $U = g(Y_D)$ and $V = t_H + t_{\texttt{net}}$, so that $W = T'_\delta$. The parameters are set as

$$a = \frac{2r}{\rho^2} \quad A = \delta\rho - t_S \quad \mu_Y = \mu \quad \sigma_Y^2 = \sigma^2$$

and (6.6) affirms that

$$\Pr[T'_\delta \leq \tau] \leq \frac{\rho}{\sqrt{4r\sigma^2 + \rho^2}} \exp\left[-\frac{2r\left((\delta\rho - t_S + \mu) - \tau\right)^2}{4r\sigma^2 + \rho^2}\right]$$

Finally, we get

$$p_{\texttt{fn}} = \Pr[T_{\delta_{\max}} \leq \tau] \leq \Pr[T'_{\delta_{\max}} \leq \tau] \leq \frac{\rho}{\sqrt{4r\sigma^2(r) + \rho^2}} \cdot \exp\left[-\frac{2r\left((\delta_{\max}\rho - t_S + \mu(r)) - \tau\right)^2}{4r\sigma^2(r) + \rho^2}\right] \tag{6.8}$$

□

## Proof of statement (b)

$\mathcal{U}$ wants to choose $\tau$ and $r$ such that the system

$$\begin{cases} p_{\texttt{fp}} \leq p_{\texttt{fp}\,\max} \\ p_{\texttt{fn}} \leq p_{\texttt{fn}\,\max} \end{cases} \tag{6.9}$$

is satisfied. Thanks to statement (a), for the first inequality, $\mathcal{U}$ imposes

$$p_{\texttt{fp}\,\max} \geq 1 - \Phi\left(\frac{\tau - rt_{\texttt{RAM}} - \mu(r)}{\sigma(r)}\right) \tag{6.10}$$

which is equivalent to

$$\tau \geq \tau_{\min}(r) \tag{6.11}$$

if we impose

$$\tau_{\min}(r) = rt_{\texttt{RAM}} + \sigma(r)\Phi^{-1}(1 - p_{\texttt{fp}\,\max}) + \mu(r)$$

Again from statement (a), for the second inequality, $\mathcal{U}$ imposes

$$p_{\texttt{fn}\,\max} \geq \frac{\rho}{\sqrt{4r\sigma^2(r) + \rho^2}} \cdot \exp\left[-\frac{2r\left((\delta_{\max}\rho - t_S + \mu(r)) - \tau\right)^2}{4r\sigma^2(r) + \rho^2}\right]$$

which is equivalent to

$$\left((\delta_{\max}\rho - t_S + \mu(r)) - \tau\right)^2 \geq \frac{4r\sigma^2(r) + \rho^2}{2r} \cdot \ln\left(\frac{\rho}{p_{\texttt{fn}\,\max}\sqrt{4r\sigma^2(r) + \rho^2}}\right) \tag{6.12}$$

Under the hypothesis $\tau < \mathbb{E}[T_{\delta_{\max}}]$, (6.12) is satisfied if

$$\tau \leq \tau_{\max}(r) \tag{6.13}$$

if we impose

$$\tau_{\max}(r) = (\delta_{\max}\rho - t_S + \mu(r)) - \sqrt{\frac{4r\sigma^2(r) + \rho^2}{2r} \cdot \ln\left(\frac{\rho}{p_{\mathtt{fn}\max}\sqrt{4r\sigma^2(r) + \rho^2}}\right)}$$

As a consequence, if $r$ is set such that $\tau_{\max}(r) \geq \tau_{\min}(r)$, any $\tau \in [\tau_{\min}(r), \tau_{\max}(r)]$ ensures that system (6.9) is satisfied. □

## 6.4 Real-life Applications

In this section we show some examples of the application of our scheme in real-life, discussing how the main design parameters affect the effectiveness of the protocol. First, we need to identify realistic values for all system parameters that are not tunable by $\mathcal{U}$: the hash computation time $t_H$, the network delay $t_{\mathtt{net}}$, and the parameters $t_S$, $t_R$ and $t_{\mathtt{RAM}}$ describing the performances of the storage media used by $\mathcal{P}$.

As shown in Section 5.4, $t_H$ can be accurately approximated by a Normal variable, of mean $\mu_H(r) = 2.2r\,\text{ms} = 2.2r \times 10^{-3}\,\text{s}$, and deviation $\sigma_H(r) = 2.5r\,\mu\text{s} = 2.5r \times 10^{-6}\,\text{s}$. The precise distribution of $t_{\mathtt{net}}$ varies according to the specific locations of the client and the server, but the experiments of Section 5.4.2 underline how such differences are minimal. As a conservative assumption, we take the maximum mean and deviation among those measured for all pinging locations, that are, $\mu_{\mathtt{net}} = 335\,\text{ms} = 3.35 \times 10^{-1}\,\text{s}$, and $\sigma_{\mathtt{net}} = 15\,\text{ms} = 1.5 \times 10^{-2}\,\text{s}$.

The HDDs average seek time $t_S$, and the times $t_R$ and $t_{\mathtt{RAM}}$ to read a 512KB block from disk and from RAM, respectively, depend on the specific model, but their order of magnitude is typically invariant. To make the analysis coherent with the extensive simulations presented in Section 6.5, we set $t_S$, $t_R$ and $t_{\mathtt{RAM}}$ according to the specifics of the hardware used to run the simulations. The HDD model used is the Hitachi Ultrastar 15K300 SAS, for which $t_S = 3.4\,\text{ms} = 3.4 \times 10^{-3}\,\text{s}$, and $t_R = 4.7\,\text{ms} = 4.7 \times 10^{-3}\,\text{s}$. The RAM model used is the Kingston DDR2-667 PC2-5300, whose access time can be estimated as AT= 45 ns, so that $t_{\mathtt{RAM}} = 5.76\,\mu\text{s} = 5.76 \times 10^{-6}\,\text{s}$.[5]

In Section 6.2.2, and in particular in Theorem 6.2.1, we showed how $\mathcal{U}$ can design the test so as to concurrently bound the false positive probability below $p_{\mathtt{fp}\max}$ and the false negative probability below $p_{\mathtt{fn}\max}$. In particular, once $\mathcal{U}$ chose $r$ such that the necessary condition $\tau_{\max}(r) - \tau_{\min}(r) \geq 0$ holds, $\mathcal{U}$ can set the threshold $\tau$ of the test as any $\tau \in [\tau_{\min}(r), \tau_{\max}(r)]$.

Even plugging into the expressions of $\tau_{\min}(r)$ and $\tau_{\max}(r)$ the realistic values identified before for all system parameters, the general dependency of such bounds on $r$, $p_{\mathtt{fp}\max}$, and $p_{\mathtt{fn}\max}$ remains hardly intelligible. For this reason, in the remainder of this section we analyze several possible

[5]All values are computed based on the discussion presented in Section 5.3, and on the specifics of the hardware used (*e.g.*, 850 sectors per track on average).

scenarios, where we fix the number of HDDs $N$ and the threshold $\delta_{\max}$, and we assume that a symmetrical false positive and false negative probability $p$ is admitted. Plotting the extrema $\tau_{\min}$ and $\tau_{\max}$ as $p$ and $r$ vary, we highlight how $r$ can be chosen once $p$ is fixed, so as to ensure that $p_{\mathtt{fn}}$ and $p_{\mathtt{fp}}$ are both bounded by $p$. If $r$ is fixed as the minimum size able to satisfy the latter requirement, the value of the threshold $\tau$ that must be used for the statistical hypothesis test is automatically and univocally determined.

However, before we discuss specific examples, let us exhibit some general considerations. Intuitively, we want the distance $\Delta = \mathbb{E}[t_{\mathtt{ret}}(\delta_{\max})] - rt_{\mathtt{RAM}}$ between the expected value of $T_{\delta_{\max}}$ and $T_{\mathtt{hon}}$ to be sufficiently large with respect to the standard deviation $\sigma(r)$ of $T_{\mathtt{hon}}$. We can therefore focus on the ratio

$$\Theta(r) = \frac{\Delta}{\sigma(r)} \approx 0.54r\frac{\delta_{\max}}{N}$$

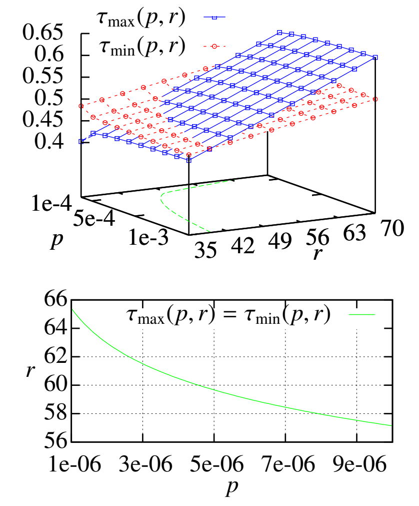
and consequently infer that the order of magnitude of $r$ must be $10N\delta_{\max}^{-1}$. Observe that, for realistic values of $N$ and $\delta_{\max}$, a similar value for $r$ is very reasonable. For instance, if $\mathcal{P}$ has $N = 10$ HDDs storing a whole copy of the file $F$, $\mathcal{U}$ can detect if $\mathcal{P}$ is storing more than $\delta_{\max} = 0.1$ of $F$ only on disk, by setting $r \approx 1000$. This corresponds to any challenge/response iteration involving 512MB of data.

**Case 1: $N = 1$, $\delta_{\max} = 0.5$.** Let us start from a scenario very favorable to $\mathcal{U}$. In this case, since $N\delta_{\max}^{-1} = 2$, we expect a challenge composed of some tents of blocks to be sufficient to detect that $\mathcal{P}$ is dishonest. Fig. 6.3a confirms this intuition is indeed true: $r \approx 60$ is sufficient to make the false positive and false negative rates both negligible ($\approx 4.5 \times 10^{-5}$). The amount of data involved in the response computation is only $\approx$ 30MB, and the threshold $\tau$ can be set as approximately half a second.

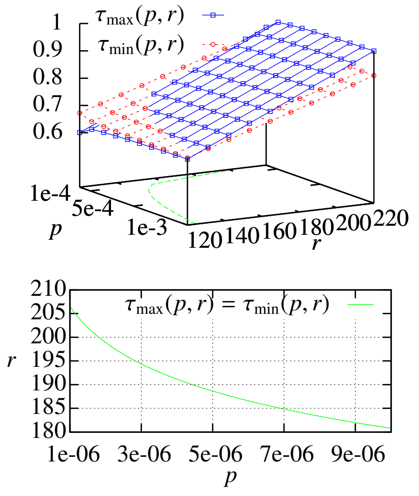
**Case 2: $N = 2$, $\delta_{\max} = 0.3$.** In this case, $N\delta_{\max}^{-1} \approx 6.7$, so we expect to need a challenge approximately three times larger than the previous case to detect that $\mathcal{P}$ is dishonest. Fig. 6.3b once again confirms what we expect: $r \approx 190$ is necessary to obtain the same confidence ($p_{\mathtt{fn}}$ and $p_{\mathtt{fp}}$ both $\approx 4.5 \times 10^{-5}$) about the reliability of our scheme. In this case, the amount of data involved in the response computation is $\approx$ 95MB, and the threshold $\tau$ must be set as approximately 0.7 s.

**Case 3: $N = 4$, $\delta_{\max} = 0.3$.** With $N\delta_{\max}^{-1} \approx 13.4$, we expect to need a challenge approximately twice as large as in Case 2. Even in this case, our prediction is confirmed by the plots, shown in Fig. 6.3c. $r \approx 320$ is necessary to obtain error rates of $\approx 4.5 \times 10^{-5}$. This means to involve $\approx$ 160MB of data in each challenge/response iteration, and that the threshold $\tau$ must be set as approximately 0.9 s.

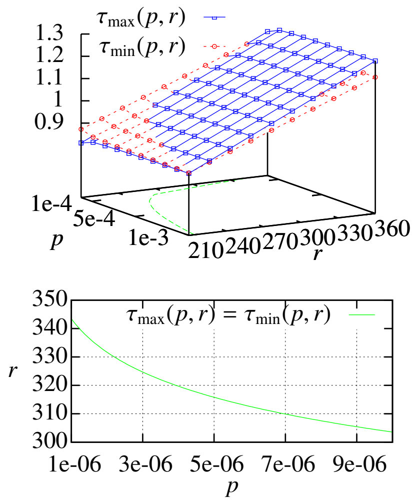
**Case 4: $N = 8$, $\delta_{\max} = 0.5$.** Since, with respect to Case 3, we increased $N$ and (almost) proportionally increased $\delta_{\max}$, we need approximately the same $r$, as shown in Fig. 6.3d, and as expected. More precisely, $r \approx 325$ is necessary to obtain error rates of $\approx 4.5 \times 10^{-5}$, and



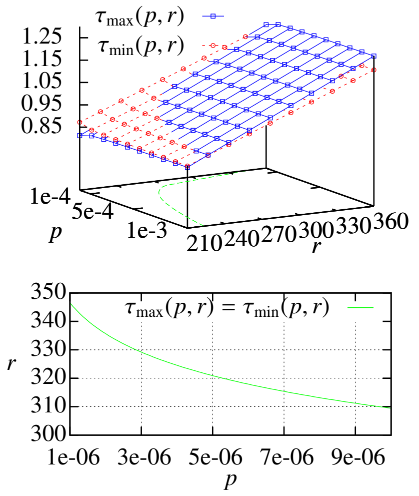
(a) Case 1: $N = 1$, $\delta_{\max} = 0.5$.



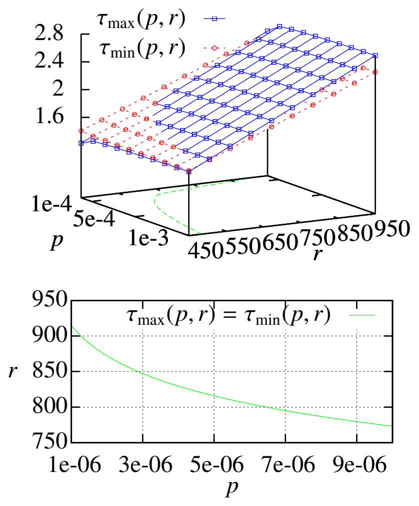
(b) Case 2: $N = 2$, $\delta_{\max} = 0.3$.



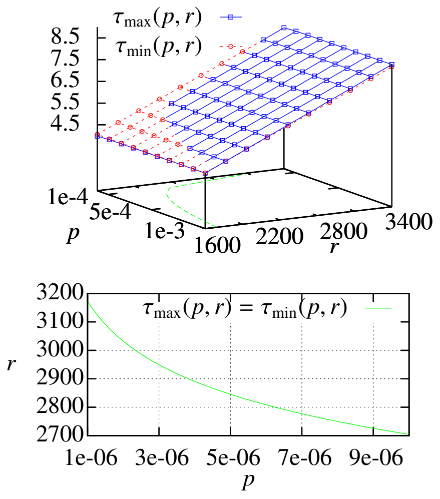
(c) Case 3: $N = 4$, $\delta_{\max} = 0.3$.



(d) Case 4: $N = 8$, $\delta_{\max} = 0.5$.



(e) Case 5: $N = 1$, $\delta_{\max} = 0.1$.



(f) Case 6: $N = 8$, $\delta_{\max} = 0.1$.

FIGURE 6.3: Real-life applications examples. In all cases: the upper plot shows $\tau_{\max}(p,r)$ and $\tau_{\min}(p,r)$, where $p$ is the maximum accepted error rate, and $r$ is the size of the challenge; the lower plot shows a zoom on the points $(p,r)$ such that $\tau_{\max}(p,r) = \tau_{\min}(p,r)$.

$\approx$ 165MB of data are involved in each challenge/response. Also in this case, the threshold $\tau$ must be set as approximately 0.9 s.

**Case 5:** $N = 1$**,** $\delta_{\max} = 0.1$**.** In this case, since $N\delta_{\max}^{-1} = 10$ is a middle ground between Case 2 and Case 3, we expected $r \approx 250$ to be sufficient to detect $\mathcal{P}$ with false positive and false negative rates $\approx 4.5 \times 10^{-5}$. On the contrary, Fig. 6.3e shows that a much larger challenge composed of $r \approx 800$ blocks is required. The rationale we identified is that decreasing $N$ and $\delta_{\max}$ increases the variance of the time $T_{\delta_{\max}}$ needed by a $\delta_{\max}$-malicious provider to return the response, thus making more difficult to upper bound $p_{\mathtt{fn}}$. It is worth to underline how this means that $\delta_{\max}$ has a *heavier* impact on the scheme than $N$: it is harder to detect $\mathcal{P}$ if it behaves more honestly, rather than if it (proportionally) increases the number of copies of $F$ available on disk. Finally, this scenario requires $\mathcal{P}$ to compute the response over $\approx$ 400MB of data, and $\mathcal{U}$ to set $\tau$ as approximately 1.5 s.

**Case 6:** $N = 8$**,** $\delta_{\max} = 0.1$**.** Even if this case is very challenging for $\mathcal{U}$, Fig. 6.3f shows that our scheme is definitely feasible, though a bit more resource-demanding. In particular, $r \approx 2900$ is necessary to obtain the same error rates of the previous cases ($\approx 4.5 \times 10^{-5}$). Accordingly, the amount of data involved in the response computation becomes $\approx$ 1450MB, which is not particularly large for a high-performance server. Finally, in this case the threshold $\tau$ must be set as approximately 6 s.

The purpose of this discussion was to show in detail how the scheme can be implemented so as to infer the behavior of the provider, keeping the false positive and false negative rates under the desired thresholds. To further help the reader to capture the effectiveness of our scheme, in the next section we will exhibit more plots, showing how the expectation of $T_\delta$ varies as a function of $\delta$ and $N$, for $r = 3000$ fixed. A comparison of the expectations of $T_\delta$ and $T_{\mathrm{hon}}$, and of the different order of magnitude of the corresponding standard deviations, immediately shows why the scheme is indeed very effective.

## 6.5 Experimental Results

To further validate our PSM protocol, we implemented a prototype consisting of a client and server program implemented in C. Since the purpose of the experiments is mainly to verify the viability of the scheme, we fixed $r = 3000$, which is sufficiently large even for *difficult* cases. For the purpose of our experiments, we created a 10GB file $F$ filled with random data, which therefore consists of 20480 512KB data blocks, numbered from 0 to 20479. We first run experiments of the PSM protocol with an honest server (*i.e.*, $\delta$=0). Afterwards, we run experiments of the PSM protocol with a malicious provider, using two levels ($\delta$=0.3 and $\delta$=0.5) of dishonesty. For a malicious provider, we also let the number of HDDs $N$ vary in $\{1, 2, 4, 8\}$.

Finally, even though it is not explicitly considered in our theoretical framework, we needed to take into account a common feature of modern operating systems (OSs) (*e.g.*, Linux and Microsoft Windows) called *file system caching*. File system caching is an integral part of the OS, used to boost read performances, and cannot be disabled. This feature makes educated guesses about the next couple of blocks a program might read, and preloads these blocks into the RAM. Tuning the parameter $CR$ in our simulations, we experimentally evaluated its influence on the response time. More specifically, we compared the case $CR$=0%, or Direct Input Output (DIO), where caching is not used, to the cases $CR$=5%, 10%, 15%, 100%. $CR$=x% means that $\mathcal{P}$'s HDDs can only cache x% of the file.

Using the configuration options discussed above (i.e., $\delta$, $N$, and $CR$), there are one possible configuration for the honest case, and 2 ($\delta$) $\times$ 4 ($N$) $\times$ 5 ($CR$) = 40 possible configurations for the malicious case. Each experiment was run once from each client location (i.e., Northern-Virginia, Sao-Paulo, Sydney, and Tokyo), summing up to 4 experiments for the honest configuration and 160 for the dishonest configurations. Finally, for each of such configurations, we performed 1000 independent runs of the PSM challenge/response scheme, and computed the average response time over the 1000 runs.

### 6.5.1 Experimental Settings

Before entering the details of the experiments performed to support the theoretical analysis, we need to clarify the experimental settings, that is, to provide the specifications of the server and clients used in the experiments.

**Server** The server we used has two 64-bit Intel Xeon E5345 processors. Each one of these processors has 4 cores, runs at a 2.33 GHz clock speed with a 1333 MHz front side bus, and has 8 MB L2 cache. The server has 16 GB DDR2-667 PC2-5300 fully buffered and registered ECC RAM installed. The installed RAM has a clock cycle of 3 nanoseconds and a 5-5-5 timing.[6] Moreover, the server comes equipped with eight Hitachi Ultrastar 15K300 SAS (Serially Attached SCSI) hard disks drives. This HDD model is characterized by 147 GB capacity and 16 MB of cache, while its performance specifications are summarized in Table 5.1. These HDDs are connected to the system via a LSI MegaRAID SAS 8480E[7] based smart array disk controller. Each one of the eight installed hard disk drives contains an ext4 file system with a default 4KB block size. Furthermore, the server runs Ubuntu Server 12.0.4 GNU/Linux.

[6] A 5-5-5 timing means that 45 nanoseconds (*i.e.*, 15 clock cycles) elapse between issueing the request for a piece of data and actually getting it.

[7] http://www.lsi.com/downloads/Public/Obsolete/Obsolete%20Common%20Files/mr_sas_stor_ug.pdf

**Clients** We use four clients for our experiments, and each one of these clients is a Virtual Machine provided by the Amazon Elastic Cloud 2 (EC2). Each client has 1 64-bit core, 600 MB of RAM, and runs Amazon Linux AMI 2013.03.1. Furthermore, each client is located in a datacenter site at a different geographic location. More specifically, we have one client at each of the following Amazon datacenter sites: (i) Northern-Virginia, (ii) Sao-Paulo, (iii) Sydney, and (iv) Tokyo.

### 6.5.2 Experimental Results with an Honest Server

The experimental results with an honest server, shown in Fig. 6.4, exhibit an average response time of approximately 7 s for all four clients at different geographic locations. Small differences can be observed among different clients, which can be attributed to differences in the corresponding network delay time. To clarify, let us have a closer look at Fig. 6.4, compared with Fig. 5.4. The difference in the average response time between the clients in Tokyo and Northern-Virginia is 209.168 ms, and the difference in the weekly average network delay to Milan between the same sites is 197.735 ms. This confirms that the difference in the average response time is mainly due to the difference in the values of $t_{\mathtt{net}}$.

Let us observe that the statistical deviation of the response time is very small. Clients in Sao-Paulo and Tokyo show a definitely negligible deviation of 0.0065118847 s and 0.0058800772 s, respectively. Even in the case of the client located in Sydney, which exhibits the largest deviation, this value is 0.117622897 s (*i.e.*, 117.622 ms), that is, an order of magnitude smaller than the average response time. In other words, the distribution of the response time for an honest provider is very concentrated around its mean, and the average measured response time can be used as basis to detect dishonest providers.

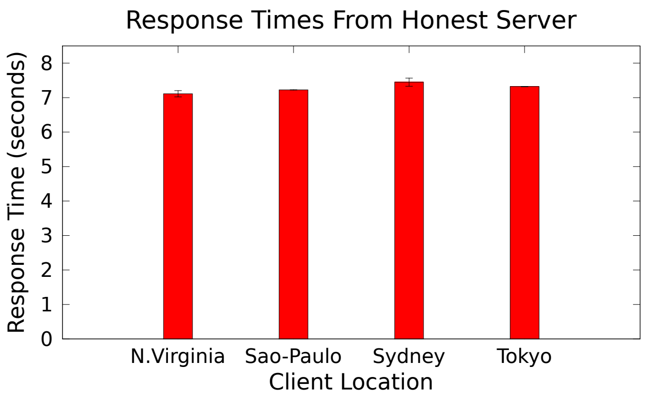


FIGURE 6.4: Average measured response time from an honest server.

### 6.5.3 Experimental Results with a Dishonest Server

Fig. 6.5 describes the average response time for the dishonest configurations, allowing a direct comparison of our theoretical predictions (Figs. 6.5a and 6.5b) and our experimental measurements (Figs. 6.5c, 6.5d, 6.5e, 6.5f). In the previous sections we described several strategies that a malicious provider can adopt to reduce the response time and try to fool the user. However, our analysis, confirmed by the experiments, ensure that even the most advantageous combination of parameters cannot prevent a dishonest provider from being detected. To see why, we need to discuss Fig. 6.5 more in detail.

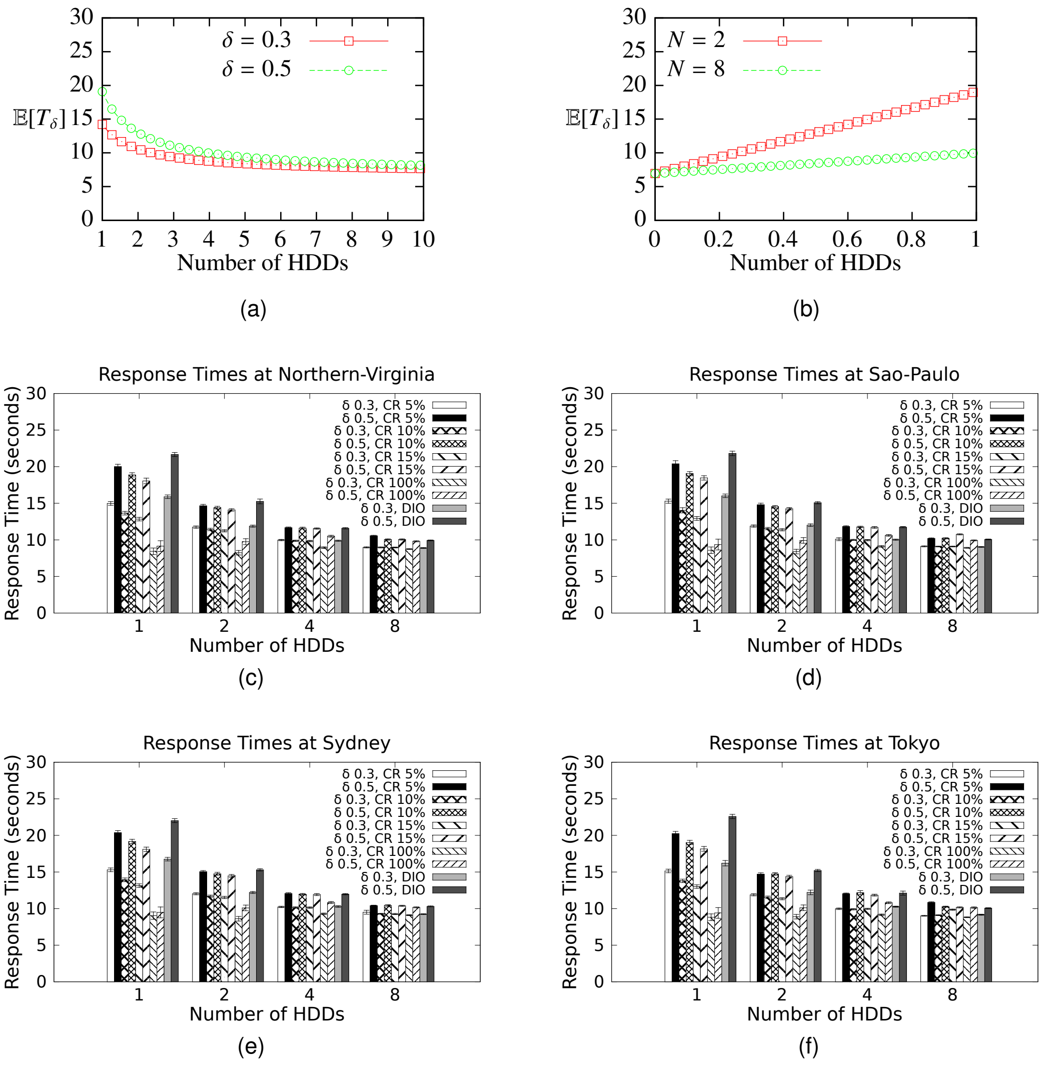


FIGURE 6.5: Average response time: comparison of theoretical predictions and experimental measurements.

For each of the four client locations, we exhibit a plot of the experimental measurements with ten bars for each number of HDDs. Each bar represents the average response time measured in correspondence of a specific combination of $\delta$ and $CR$. First, the plots confirm that (for any specific configuration) the response times measured from a 0.3-malicious provider are indeed lower than those for a 0.5-malicious provider. For instance, Fig. 6.5c shows that the client at Northern-Virginia, if $CR$=5% and $N = 1$, experiences an increase in the average response time from 14.9803 s to 20.0137 s when $\delta$ passes from 0.3 to 0.5. Similarly, increasing the number of HDDs used results in reduced response times, but the benefit of adding more HDDs decreases with the number of HDDs. To properly evaluate the influence of the number of HDDs, we consider the simulation results corresponding to $CR$=0%, which is the only case when we can isolate the impact of $N$. If we focus on the simulation results of the client in Sydney (Fig. 6.5e), for $\delta$=0.3 and $CR$=0%, we see that the response time is 16.7557 s for $N$=1, 12.2033 s for $N$=2, 10.2683 s for $N$=4, and 9.2432 s for $N$=8, with a sub-linear drop. Finally, increasing the cache size does have the intended effect of boosting read performances, hence reducing the average response time. For instance, focusing on the client in Sao-Paulo, from Fig. 6.5d we see that, for $N$=1 and $\delta$=0.3, the average response time is 15.2866 s for $CR$=5%, 14.0356 s for $CR$=10%, 12.9302 s for $CR$=15%, and 8.5798 s for $CR$=100%.

However, in all the configurations considered, a malicious provider can be detected even if it combines all the previous strategies. To show why, let us focus on the case of a client located in Sidney, comparing Figs. 6.4 and 6.5e. The client in Sidney is the most vulnerable, since it shows both the highest average response time in the honest configuration, and the highest standard deviation. However, we can easily see that this is not enough to conceal a cheating provider. The average response time for Sidney is 7.4479 s if the provider is honest, and at least 8.6133 s (using the best configuration: $\delta$=0.3, $N$=2, and $CR$=100%) if the provider is malicious. The difference between the two average response times is at least 1.1654 s, which is *9.8 times the standard deviation*. This means that the two cases are easily statistically distinguishable.

Finally, let us focus on the theoretical plots. Figs. 6.5a and 6.5b show the approximation

$$\mathbb{E}[T_\delta] \approx \frac{\delta r}{N}(t_R + t_S) + \mu(r) = \left(\frac{\delta}{N} \cdot 24.3 + 6.935\right)\text{s}$$

of the average response time, obtained (according to Section 6.4) setting $r = 3000$, $t_R = 4.7$ ms, $t_S = 3.4$ ms, and $\mu(r) = (2.2r + 335)$ ms. In particular, Fig. 6.5a shows the plot as a function of $N$, for both $\delta = 0.3$ and $\delta = 0.5$, while Fig. 6.5b shows the plot as a function of $\delta$, for $N = 2$ and $N = 8$. As the reader can check, our theoretical predictions almost perfectly match the simulation results. The trend of the predicted and measured response times is the same, with experimental results for all four client locations being 1 to 2 s slower than the expected response times.

Such a difference can be explained considering that our theoretical framework necessarily fails to capture some very technological aspects. However, the fact that simulated dishonest providers are *slower* than our prediction confirms that our model is *more conservative* than reality. In particular, we assumed that a block is always read within a specified amount of time, while this is not the case in many practical settings. In fact, we did not take into account the processing delay introduced by code within the kernel that must be executed between the moment a program makes a read system call, and the moment the actual read takes place. The model does not even consider the processing delays due to several other software (*e.g.*, processing delays introduced by device drivers and firmware) and hardware components (*e.g.*, transmission delay over several shared buses, and processing delay introduced by controllers) on the execution path of a read system call. Summing up, the almost constant time difference observed between the estimated and measured response times is caused by several processing delays that, if taken into account, would have made our theoretical model too hard to handle.

## 6.6 Summary

In this chapter, we addressed the problem of verifying the compliance of outsourced storage service providers to the Service Level Agreement (SLA) stipulated with the client. In particular, we focused on verifying (probabilistically) whether the client's data are indeed stored on the agreed storage medium. In Chapter 5, we defined this problem as Provable Storage Medium (PSM), which is a concept more restrictive than the one of Provable Data Possession (PDP), that only aims at verifying the integrity of the data at the server side.

We proposed a PSM scheme where, based on a statistical hypothesis test, the provider is considered dishonest if it fails to provide the response to a data request within a given time constraint. A detailed analysis of the involved variables (network delay, disk and RAM access time) allows us to accurately predict the false positive and false negative rates of the test. We also run an extensive experimental campaign, whose results support the correctness of our analysis, as well as the viability of our proposal.

# 7 Conclusions

In this thesis, we presented two main contributions provided by the candidate during his PhD studies. We addressed two topics of primary importance in the security community: Distributed Sensor Network (DSN) and Cloud Storage (CS).

DSN is a model for environmental sensing and monitoring, characterized by three main assumptions: (i) the network is unattended and relies on distributed functionalities, (ii) sensors are free to randomly roam in the environment, and (iii) sensors have limited storage, computational and energetic resources. Similar features make the design of a secure and efficient data handling scheme at the same time extremely important and particularly challenging. Being able to identify a reliable and practical framework means to address at the same time the requirements of a wide range of military, civilian and industrial applications.

CS is recently establishing itself as the reference paradigm for data storage. Everyone, from big companies to private users, currently relies on remote storage services for her most important data, especially because CS is generally considered very reliable. However, the compliance of service providers to the Service Level Agreement (SLA) should not be given for granted, as shown by the considerable attention raised in the literature by the problem of Provable Data Possession (PDP). In a world where the pace of life is becoming ever faster, data access time becomes a primary concern. With this in mind, PDP cannot be the only concern considered when stipulating a SLA: minimum performances of the storage media used in the cloud must be stated, and therefore compliance to this novel aspect of SLAs must be remotely verifiable.

## 7.1 Contributions and Future Work – Part I

In Part I, we presented a survey of security issues and countermeasures for DSNs, and proposed a solution able to provide, at once, all the security requirements of this type of networks. The survey is itself of remarkable interest, especially because it concerns very recent technologies and research work, which, at least partially, still needed to be assembled in a unified body. However, the main contributions provided by Part I can be found in Chapter 3, and can be summarized as follows:

**A** We introduced a very energy-efficient scheme that provides data availability and confidentiality *without relying on any cryptographic primitive*, but only on local secret sharing and nodes mobility.

**B** We provided analytic bounds able to accurately predict, under precise assumptions, the amount of sensed data accessible by both the sink and the adversary as a function of their capabilities and of system parameters.

**C** We discussed the assumptions used to derive our bounds, showing that their correctness is only a matter of time: the faster information gets diffused by nodes mobility, the earlier system performances become predictable. Since information sharing improves diffusion, high network density can compensate for slowly moving sensors.

**D** We introduced metrics able to describe to which extent our scheme can enforce data integrity and source-location privacy, based on the ability of the sink to detect fake data injected by the adversary, and on the chances of the adversary to correctly infer the origin of some sensed data.

**E** We ran an extensive simulation campaign, which broadly confirms the accuracy of our analysis, though providing some remarkable insights on the relevance of information diffusion, and on the limited dependency of our theoretical predictions on the application setting.

Altogether, our solution represents at the same time: (i) a proof of concept for the fact that a proper information handling and mobility leveraging can provide security for DSNs, and (ii) a general framework suited for a lot of different scenarios, which can be easily adapted to the specific features of the network under analysis. We provide precise indications about how the parameters of the sharing scheme should be chosen according to the density and mobility degree of the network, to the capabilities of the sink and the adversary and to the desired trade-off between different security requirements.

We envisage two main lines to extend our work in the next future:

- We highlighted how the mobility model of the network plays a fundamental role in answering whether we can predict the behavior of the network, and to which extent. A more general analysis, able to merge the contributions of Sections 3.3 and 3.4, would allow to quantify the level of confidentiality and availability ensured by information sharing and diffusion, in a comprehensive way that considers, at the same time, the impact of all relevant parameters, such as: the number of shares generated from each piece of information, and those necessary to recover it; the amount of nodes accessed; the density of the network, in respect to the communication range; the speed and randomness of the sensors movements; etc. A similar study requires a profound knowledge of the theory of stochastic processes and, more generally, a considerable familiarity with advanced probabilistic tools.

- We considered secret sharing as the most straightforward approach to concurrently provide confidentiality and reliability, in particular when the shares can be diffused efficiently leveraging sensors movements. However, there is a whole body of knowledge about error and erasure correcting codes, that can be rightfully expected to perform much better than secret sharing. In fact, if we associate erasures with data loss, and errors with fake-data injection, any combination of data integrity, availability and confidentiality can be *provably* obtained simply choosing the proper code. Based on the rationale that data routing should be avoided (or, at least, minimized), we believe that the most promising approach would be to use Low Density Parity Check (LDPC) codes, with parameters induced by the temporary network topology, so as to be able to exchange parity bits only locally. To this end, a detailed analysis of the mobility model of the network is once again necessary. In fact, predicting how the topology of the network evolves allows to understand which parity bits a sensors is expected to hold. This information is crucial, since the main source of data loss or poisoning are captured sensors, and a single captured sensor may generate a huge amount of corrupted data.

## 7.2 Contributions and Future Work – Part II

Part II was dedicated to the problematics introduced by remotely storing private data on the cloud. In Chapter 4, we discussed the main requirement in this context: to enable any user to verify the integrity of her data, while locally storing little or no information at all. This property, usually referred to as Provable Data Possession (PDP), must be ensured while preventing any access of the provider to the clear-version of the user's data, which she might want to keep confidential. More generally, we believe that any CS infrastructure should enable its users to verify all main aspects of the SLA, not just data integrity. For this reason, in Chapters 5 and 6 we provided the following contributions:

**A** We introduced the concept of Provable Storage Medium (PSM), to denote the ability of the users to verify that the storage medium used by the provider to store their data is the one that gives at least the performances agreed in the SLA.

**B** We proposed a scheme based on time measurements for challenge/response interactions between the client and the server, able to concurrently provide PDP and PSM. The computational cost and memory utilization at the client side are very limited, and can be easily implemented in a mobile resource constrained device such as a smartphone.

**C** We performed a thorough analysis of both the system model and the possible behaviors of the provider, showing that the chances for the service provider not to comply with the SLA while being not detected can be made negligible by a proper choice of the parameters. In particular, we specified what explicit agreement about the response delivery time can be introduced in the SLA, to ensure that data are indeed stored in RAM, or on any storage medium with better access performances.

**D** We performed extensive experiments that involve real client and server instances. The experimental results do support the quality and viability of our solution.

We underline once again that what we are pursuing is not simply the answer to a security curiosity, but the solution to possible real-life issues. In fact, think to websites that require very low delays, or to real-time recovery of critical systems. In those and many other contexts, not respecting a pre-determined response time could result in remarkable financial losses or in significant security threats.

Future work could consist in both improving or generalizing our work. In particular, we identify two main research directions:

- Our PSM solution was built upon a previous PDP scheme. Even if the main idea can be righteously expected to generalize to a wide range of PDP protocols, our analysis is partially related to the specific scheme considered. In fact, we needed to quantify the time necessary to compute the proof of possession, so we considered a very efficient scheme where the proof is obtained by applying a hash function to the challenged blocks of data. However, other schemes in the literature implement other desirable properties (*e.g.*, homomorphic tags that allow the client to store a fixed amount of data, independent from the number of possible challenges [146]), at the cost of a slightly larger computational burden at the server side. It would be appreciable to generalize our protocol so as to make it adjustable to any PDP scheme. Alternatively, it would be of secure interest to formally introduce PSM among the desiderata of PDP schemes: if we identify the maximum computational time of a PDP proof that allows to concurrently address PSM, we would be able to state how PSM can be traded-off against other requirements of a CS application.

- Our work opens a new perspective towards SLA compliance in CS systems. Previous works (almost) only considered the problem of checking the integrity of the remotely stored file. However, there are other clauses than can be included in the SLA, and that the user may be therefore willing to verify. Among them, the most interesting are probably the number of copies of the file, and the location of the server(s) that store them, because both may affect the time to retrieve data, the likelihood of server failures, and the resilience to such an eventuality. The former property was briefly discussed in [167], which however completely lacks a general and theoretical framework. A possible and interesting line of research is therefore to investigate such issues more deeply.

# Bibliography

[1] R. Di Pietro, S. Guarino, N. V. Verde, and J. Domingo-Ferrer. Security in wireless ad-hoc networks – a survey. *Submitted to the Elsevier Computer Communications*, 2014.

[2] R. Di Pietro and S. Guarino. Confidentiality and availability issues in mobile unattended wireless sensor networks. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*, pages 1–6, June 2013.

[3] R. Di Pietro and S. Guarino. Data confidentiality and availability via secret sharing and node mobility in uwsn. In *INFOCOM: The 32nd IEEE International Conference on Computer Communications*, 2013.

[4] R. Di Pietro and N. V. Verde. Epidemic data survivability in unattended wireless sensor networks. In *Proceedings of the fourth ACM conference on Wireless network security*, pages 11–22. ACM, 2011.

[5] S. Guarino, E. S. Canlar, M. Conti, R. Di Pietro, and A. Solanas. Provable storage medium for data storage outsourcing. *Submitted to the IEEE Transactions on Services Computing*, 2014.

[6] D. Chakrabarti, J. Leskovec, C. Faloutsos, S. Madden, C. Guestrin, and M. Faloutsos. Information Survival Threshold in Sensor and P2P Networks. *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pages 1316–1324, May 2007. ISSN 0743-166X.

[7] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292 – 2330, 2008. ISSN 1389-1286.

[8] IEEE. *IEEE Standard 802.15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs).* 2006.

[9] ZigBee Alliance. Zigbee specification. *ZigBee document 053474r06, version*, 1, 2005.

[10] A. Hegland, E. Winjum, S. Mjolsnes, C. Rong, O. Kure, and P. Spilling. A survey of key management in ad hoc networks. *Communications Surveys Tutorials, IEEE*, 8(3):48–66, rd 2006. ISSN 1553-877X.

[11] E. Cayirci and C. Rong. *Security in Wireless Ad Hoc and Sensor Networks*. Wiley; 1 edition, 2009.

[12] H. Yih-Chun and A. Perrig. A survey of secure wireless ad hoc routing. *IEEE Security & Privacy Magazine*, 2(3):28–39, May 2004. ISSN 1540-7993.

[13] C. Sreedhar, S. M. Verma, and P. N. Kasiviswanath. A Survey on Security Issues in Wireless Ad hoc Network Routing Protocols. *International Journal*, 02(02):224–232, 2010.

[14] A. Mpitziopoulos and D. Gavalas. A survey on jamming attacks and countermeasures in WSNs. *Surveys &amp; Tutorials*, 11(4):42–56, 2009. ISSN 1553-877X.

[15] R. Pickholtz, D. Schilling, and L. Milstein. Theory of Spread-Spectrum Communications–A Tutorial. *IEEE Transactions on Communications*, 30(5):855–884, May 1982. ISSN 0096-2244.

[16] I. Oppermann, L. Stoica, a. Rabbachin, Z. Shelby, and J. Haapola. UWB wireless sensor networks: UWEN - a practical example. *IEEE Communications Magazine*, 42(12):S27–S32, December 2004. ISSN 0163-6804.

[17] W. L. Stutzman and G. A. Thiele. *Antenna Theory and Design*. New York : J. Wiley, 2 edition, 1997.

[18] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing - MobiHoc '05*, page 46, 2005.

[19] A. D. Wood, J. a. Stankovic, and G. Zhou. DEEJAM: Defeating Energy-Efficient Jamming in IEEE 802.15.4-based Wireless Networks. *2007 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 60–69, June 2007.

[20] A. D. Wood, J. A. Stankovic, and S. H. Son. Jam: A jammed-area mapping service for sensor networks. In *Proceedings of the 24th IEEE International Real-Time Systems Symposium*, RTSS '03, pages 286–, Washington, DC, USA, 2003. IEEE Computer Society.

[21] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou. JAID: An algorithm for data fusion and jamming avoidance on distributed sensor networks. *Pervasive and Mobile Computing*, 5(2):135–147, April 2009. ISSN 15741192.

[22] O. Kommerling and M. Kuhn. Design principles for tamper-resistant smartcard processors. In *of the USENIX Workshop on Smartcard*, pages 9–20, Chicago, Illinois, 1999. USENIX Association.

[23] R. J. Anderson and M. G. Kuhn. Low cost attacks on tamper resistant devices. In *Proceedings of the 5th International Workshop on Security Protocols*, pages 125–136, London, UK, 1998. Springer-Verlag.

[24] R. Anderson and M. G. Kuhn. Tamper resistance: a cautionary note. In *In Proceedings of the Second Usenix Workshop on Electronic Commerce*, pages 1–11, Oakland, California, 1996. USENIX Association.

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

[26] A. Becher, Z. Benenson, and M. Dornseif. Tampering with motes: Real-world physical attacks on wireless sensor networks. *Security in Pervasive Computing*, pages 104–118, 2006.

[27] Y. Law, P. Hartel, J. den Hartog, and P. Havinga. Link-layer jamming attacks on S-MAC. In *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*, pages 217–225. IEEE, 2004.

[28] A. Wood and J. 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.

[29] F. Stajano and R. J. Anderson. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In *Proceedings of the 7th International Workshop on Security Protocols*, pages 172–194, London, UK, 2000. Springer-Verlag.

[30] T. Martin, M. Hsiao, and J. Krishnaswami. Denial-of-service attacks on battery-powered mobile computers. In *Second IEEE Annual Conference on Pervasive Computing and Communications, 2004. Proceedings of the*, pages 309–318, Washington, DC, USA, 2004. IEEE Computer Society.

[31] D. Raymond, R. Marchany, M. Brownfield, and S. Midkiff. Effects of Denial-of-Sleep Attacks on Wireless Sensor Network MAC Protocols. *IEEE Transactions on Vehicular Technology*, 58(1):367–380, January 2009. ISSN 0018-9545.

[32] D. R. Raymond, R. C. Marchany, and S. F. Midkiff. Scalable, Cluster-based Anti-replay Protection for Wireless Sensor Networks. In *2007 IEEE SMC Information Assurance and Security Workshop*, pages 127–134. IEEE, June 2007.

[33] W. Zhang, N. Subramanian, and G. Wang. Lightweight and Compromise-Resilient Message Authentication in Sensor Networks. *2008 IEEE INFOCOM - The 27th Conference on Computer Communications*, pages 1418–1426, April 2008. ISSN 0743-166X.

[34] A. Perrig, R. Szewczyk, J. Tygar, V. Wen, and D. Culler. SPINS: Security protocols for sensor networks. *Wireless networks*, 8(5):521–534, 2002.

[35] 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 - MobiCom '00*, pages 255–265, New York, New York, USA, 2000. ACM Press.

[36] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 113—–127. Elsevier, 2003.

[37] V. Singh, S. Jain, and J. Singhai. Hello Flood Attack and its Countermeasures in Wireless Sensor Networks. *International Journal of Computer Science*, 7(3):23, 2010.

[38] Z. Karakehayov. Using REWARD to detect team black-hole attacks in wireless sensor networks. In *Workshop on Real-World Wireless Sensor Networks*, Stockholm, Sweden, 2005. Citeseer.

[39] E. H. Ngai, J. Liu, and M. Lyu. On the Intruder Detection for Sinkhole Attack in Wireless Sensor Networks. In *2006 IEEE International Conference on Communications*, pages 3383–3389. Ieee, 2006.

[40] D. Dallas, C. Leckie, and K. Ramamohanarao. Hop-Count Monitoring: Detecting Sinkhole Attacks in Wireless Sensor Networks. *15th IEEE International Conference on Networks*, pages 176–181, November 2007. ISSN 1556-6463.

[41] A. A. Pirzada and C. McDonald. Circumventing sinkholes and wormholes in wireless sensor networks. In *Conference on Wireless Ad Hoc Networks*, 2005.

[42] I. Krontiris, T. Dimitriou, T. Giannetsos, and M. Mpasoukos. Intrusion detection of sinkhole attacks in wireless sensor networks. *Algorithmic Aspects of Wireless Sensor Networks*, pages 150–161, 2008.

[43] Y.-C. Hu, A. Perrig, and D. Johnson. Packet leashes: a defense against wormhole attacks in wireless networks. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, volume 3, pages 1976–1986. Ieee, 2002.

[44] S. Kaplantzis, A. Shilton, N. Mani, and Y. A. Sekercioglu. Detecting Selective Forwarding Attacks in Wireless Sensor Networks using Support Vector Machines. In *3rd International Conference on Intelligent Sensors, Sensor Networks and Information*, pages 335–340. Ieee, 2007.

[45] B. Yu and B. Xiao. Detecting selective forwarding attacks in wireless sensor networks. *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*, page 8 pp., 2006.

[46] Y. Yu, R. Govindan, and D. Estrin. Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks. *UCLA Computer Science Department Technical Report, UCLA-CSD TR-01-0023*, 2001.

[47] J. R. Douceur. The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251–260, London, UK, 2002. Springer-Verlag.

[48] J. Newso, 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, New York, NY, USA, 2004. ACM.

[49] C. Wang, K. Sohraby, B. Li, M. Daneshmand, and Y. Hu. A survey of transport protocols for wireless sensor networks. *Network, IEEE*, 20(3):34–40, 2006.

[50] B. Hull, K. Jamieson, and H. Balakrishnan. Mitigating Congestion in Wireless Sensor Networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 134–147, New York, NY, USA, 2004. ACM.

[51] C. Wan, S. Eisenman, and A. Campbell. CODA: congestion detection and avoidance in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 266–279. ACM, 2003.

[52] C. Ee and R. Bajcsy. Congestion control and fairness for many-to-one routing in sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 148–161. ACM, 2004.

[53] C. Wan, S. Eisenman, A. Campbell, and J. Crowcroft. Siphon: overload traffic management using multi-radio virtual sinks in sensor networks. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 116–129. ACM, 2005.

[54] A. Woo and D. Culler. A transmission control scheme for media access in sensor networks. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 221–235. ACM, 2001.

[55] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation-Volume 1*, pages 2–2. USENIX Association, 2004.

[56] Y. Iyer, S. Gandham, and S. Venkatesan. STCP: a generic transport layer protocol for wireless sensor networks. In *Computer Communications and Networks, 2005. ICCCN 2005. Proceedings. 14th International Conference on*, pages 449–454. IEEE, 2005.

[57] Y. Sankarasubramaniam, O. Akan, and I. Akyildiz. ESRT: event-to-sink reliable transport in wireless sensor networks. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 177–188. ACM, 2003.

[58] S. Park, R. Vedantham, R. Sivakumar, and I. Akyildiz. A scalable approach for reliable downstream data delivery in wireless sensor networks. In *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pages 78–89. ACM, 2004.

[59] C. Wan, A. Campbell, and L. Krishnamurthy. PSFQ: a reliable transport protocol for wireless sensor networks. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 1–11. ACM, 2002.

[60] A. Dunkels, J. Alonso, T. Voigt, and H. Ritter. Distributed TCP caching for wireless sensor networks. Technical report, SICS Report, 2004.

[61] H. Zhang, A. Arora, and Y.-r. Choi. Reliable bursty convergecast in wireless sensor networks. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 266–276. ACM, 2007.

[62] F. Yunus, N. Ismail, S. Ariffin, A. Shahidan, N. Fisal, and S. Syed-Yusof. Proposed transport protocol for reliable data transfer in wireless sensor network (WSN). In *Modeling, Simulation and Applied Optimization (ICMSAO), 2011 4th International Conference on*, pages 1–7. IEEE, 2011.

[63] L. Lamport and R. Shostak. The Byzantine generals problem. *ACM Transactions on Programming*, 4(3):382–401, 1982.

[64] L. Buttyán and L. Csik. Security analysis of reliable transport layer protocols for wireless sensor networks. In *8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010*, pages 1–10, 2010.

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

[66] P. Ning, A. Liu, and W. Du. Mitigating DoS attacks against broadcast authentication in wireless sensor networks. *ACM Transactions on Sensor Networks*, 4(1):1–35, January 2008. ISSN 15504859.

[67] M. Ameen, J. Liu, and K. Kwak. Security and privacy issues in wireless sensor networks for healthcare applications. *Journal of Medical Systems*, 36(1):93–101, 2012. ISSN 0148-5598.

[68] M. Anand, Z. Ives, and I. Lee. Quantifying eavesdropping vulnerability in sensor networks. In *Proceedings of the 2nd international workshop on Data management for sensor networks - DMSN '05*, page 3, New York, New York, USA, 2005. ACM Press.

[69] R. Di Pietro, L. V. Mancini, and A. Mei. Energy efficient node-to-node authentication and communication confidentiality in wireless sensor networks. *Wireless Networks*, 12 (6):709–721, May 2006. ISSN 1022-0038.

[70] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 41–47, New York, New York, USA, 2002. ACM Press.

[71] A. Rasheed and R. Mahapatra. The three-tier security scheme in wireless sensor networks with mobile sinks. *Parallel and Distributed Systems, IEEE Transactions on*, 23(5):958–965, May 2012. ISSN 1045-9219.

[72] C. Castelluccia, E. Mykletun, and G. Tsudik. Efficient aggregation of encrypted data in wireless sensor networks. *The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pages 109–117, 2005.

[73] R. Di Pietro, P. Michiardi, and R. Molva. Confidentiality and integrity for data aggregation in WSN using peer monitoring. *Security and Communication Networks*, 2(2), 2009.

[74] A. Viejo, J. Domingo-Ferrer, F. Sebé, and J. Castellà-Roca. Secure many-to-one communications in wireless sensor networks. *Sensors*, 9(7):5324–5338, 2009.

[75] J. Deng, R. Han, and S. Mishra. Countermeasures against traffic analysis attacks in wireless sensor networks. In *Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks*, pages 113–126. IEEE Computer Society, 2005.

[76] A. Wadaa, S. Olariu, L. Wilson, M. Eltoweissy, and K. Jones. On providing anonymity in wireless sensor networks. In *Tenth International Conference on Parallel and Distributed Systems, 2004. ICPADS 2004.*, pages 411–418. Ieee, 2004.

[77] A. Perrig and D. Song. Random key predistribution schemes for sensor networks. *Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405)*, (April):197–213, 2003.

[78] B. Parno, A. Perrig, and V. Gligor. Distributed Detection of Node Replication Attacks in Sensor Networks. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 49–63, Washington, DC, USA, 2005. IEEE Computer Society.

[79] J. Deng. A pairwise key pre-distribution scheme for wireless sensor networks. In *Proceedings of the 10th ACM conference on Computer and communication security - CCS '03*, volume V, page 42, New York, New York, USA, 2005. The University of North Carolina at Greensboro.

[80] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *Proceedings of the 10th ACM conference on Computer and communication security - CCS '03*, page 52, New York, New York, USA, 2003. ACM Press.

[81] A. Perrig, R. Canetti, J. Tygar, and D. Song. Efficient authentication and signing of multicast streams over lossy channels. In *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*, volume 28913, pages 56–73. IEEE, 2000.

[82] R. Lu, X. Lin, H. Zhu, X. Liang, and X. Shen. Becan: A bandwidth-efficient cooperative authentication scheme for filtering injected false data in wireless sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 23(1):32–43, Jan 2012. ISSN 1045-9219.

[83] S. Roy, M. Conti, S. Setia, and S. Jajodia. Secure data aggregation in wireless sensor networks. *Information Forensics and Security, IEEE Transactions on*, 7(3):1040–1052, June 2012. ISSN 1556-6013.

[84] S. Nath, H. Yu, P. B. Gibbons, and S. Seshan. Synopsis diffusion for robust aggregation in sensor networks. In *IN SENSYS*, pages 250–262. ACM Press, 2004.

[85] W. Khan, Y. Xiang, M. Aalsalem, and Q. Arshad. Mobile phone sensing systems: A survey. *Communications Surveys Tutorials, IEEE*, 15(1):402–427, Quarter. ISSN 1553-877X.

[86] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. In *In: Workshop on World-Sensor-Web (WSW'06): Mobile Device Centric Sensor Networks and Applications*, pages 117–134, 2006.

[87] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. A. Peterson. People-centric urban sensing. In *Proceedings of the 2Nd Annual International Workshop on Wireless Internet*, WICON '06, New York, NY, USA, 2006. ACM.

[88] A. Kansal, M. Goraczko, and F. Zhao. Building a sensor network of mobile phones. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, IPSN '07, pages 547–548, New York, NY, USA, 2007. ACM.

[89] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pages 41–47, New York, NY, USA, 2002. ACM.

[90] R. Di Pietro, L. V. Mancini, C. Soriente, A. Spognardi, and G. Tsudik. Catch Me (If You Can): Data Survival in Unattended Sensor Networks. *2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 185–194, March 2008.

[91] M. Albano, S. Chessa, and R. Di Pietro. A model with applications for data survivability in critical infrastructures. *Journal of Information Assurance and Security*, 4(6):629–639, 2009.

[92] D. Ma, C. Soriente, and G. Tsudik. New adversary and new threats: security in unattended sensor networks. *IEEE Network*, 23(2):43–48, March 2009. ISSN 0890-8044.

[93] R. Merkle. A certified digital signature. In *Proceedings on Advances in cryptology*, pages 218–238, Santa Barbara, California, United States, 1989. Springer-Verlag New York, Inc.

[94] Nist. FIPS PUB 197: Announcing the Advanced Encryption Standard (AES), 2001.

[95] NIST. SKIPJACK and KEA Algorithm Specifications Version 2.0. Technical report, 1998.

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

[97] R. Di Pietro, L. V. Mancini, C. Soriente, A. Spognardi, and G. Tsudik. Data Security in Unattended Wireless Sensor Networks. *IEEE Transactions on Computers*, 58(11): 1500–1511, 2009. ISSN 0018-9340.

[98] R. Di Pietro, L. V. Mancini, C. Soriente, A. Spognardi, and G. Tsudik. Playing hide-and-seek with a focused mobile adversary in unattended wireless sensor networks. *Ad Hoc Networks*, 7(8):1463–1475, November 2009. ISSN 15708705.

[99] S. Reddy, S. Ruj, and A. Nayak. Distributed data survivability schemes in mobile unattended wireless sensor networks. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 979–984, Dec 2012.

[100] W. Ren, J. Zhao, and Y. Ren. Network Coding based Dependable and Efficient Data Survival in Unattended Wireless Sensor Networks. *Journal of Communications*, 4(11): 894–901, December 2009. ISSN 1796-2021.

[101] M. Bellare and B. Yee. Forward-security in private-key cryptography. In *Proceedings of the 2003 RSA conference on The cryptographers' track*, pages 1–18, San Francisco, CA, USA, 2003. Springer.

[102] Z. Liu, J. Ma, Y. Park, and S. Xiang. Data security in unattended wireless sensor networks with mobile sinks. *Wireless Communications and Mobile Computing*, 12(13):1131–1146, 2012. ISSN 1530-8677.

[103] D. Ma and G. Tsudik. DISH: Distributed Self-Healing. In *SSS '08: Proceedings of the 10th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 47–62, Detroit, MI, 2008. Springer-Verlag.

[104] R. Di Pietro, D. Ma, C. Soriente, and G. Tsudik. POSH: Proactive co-Operative Self-Healing in Unattended Wireless Sensor Networks. In *SRDS '08: Proceedings of the 2008 Symposium on Reliable Distributed Systems*, pages 185–194, Washington, DC, USA, 2008. IEEE Computer Society.

[105] R. Di Pietro, G. Oligeri, C. Soriente, and G. Tsudik. Intrusion-Resilience in Mobile Unattended WSNs. In *Proceedings IEEE INFOCOM*, pages 1–9, San Diego, California, USA, March 2010. IEEE Press.

[106] R. Di Pietro, G. Oligeri, C. Soriente, and G. Tsudik. Securing Mobile Unattended WSNs against a Mobile Adversary. In *29th IEEE Symposium on Reliable Distributed Systems*, pages 11–20, New Delhi, India, 2010. IEEE.

[107] D. Ma and G. Tsudik. Forward-Secure Sequential Aggregate Authentication (Short Paper). In *IEEE Symposium on Security and Privacy, S&P'07*, pages 86–91, 2007.

[108] R. Di Pietro, C. Soriente, A. Spognardi, and G. Tsudik. Collaborative authentication in unattended WSNs. In *Proceedings of the second ACM conference on Wireless network security - WiSec '09*, pages 237–244, Zürich, Switzerland, 2009. ACM Press.

[109] R. Di Pietro, C. Soriente, A. Spognardi, and G. Tsudik. Intrusion-resilient integrity in data-centric unattended WSNs. *Pervasive and Mobile Computing*, 7(4):495–508, August 2011. ISSN 15741192.

[110] Y. Ren, V. Oleshchuk, F. Y. Li, and X. Ge. Security in mobile wireless sensor networks–a survey. *Journal of Communications*, 6(2):128–142, 2011.

[111] Q. Wang, X. Wang, and X. Lin. Mobility increases the connectivity of k-hop clustered wireless networks. In *Proceedings of the 15th annual international conference on Mobile computing and networking*, pages 121–132. ACM, 2009.

[112] X. Wang, X. Wang, and J. Zhao. Impact of mobility and heterogeneity on coverage and energy consumption in wireless sensor networks. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 477–487. IEEE, 2011.

[113] S. Čapkun, J.-P. Hubaux, and L. Buttyán. Mobility helps security in ad hoc networks. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking &Amp; Computing*, MobiHoc '03, pages 46–56, New York, NY, USA, 2003. ACM.

[114] A. Dua, N. Bulusu, W.-c. Feng, and W. Hu. Towards trustworthy participatory sensing. In *HotSec'09: Proceedings of the Usenix Workshop on Hot Topics in Security*, 2009.

[115] 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, Jan 2009.

[116] M. M. Groat, B. Edwards, J. Horey, W. He, and S. Forrest. Enhancing privacy in participatory sensing applications with multidimensional data. In *PerCom*, pages 144–152, 2012.

[117] R. Zhang, J. Shi, Y. Zhang, and C. Zhang. Verifiable privacy-preserving aggregation in people-centric urban sensing systems. *Selected Areas in Communications, IEEE Journal on*, 31(9):268–278, September 2013. ISSN 0733-8716.

[118] K. L. Huang, S. S. Kanhere, and W. Hu. Preserving privacy in participatory sensing systems. *Computer Communications*, 33(11):1266 – 1280, 2010. ISSN 0140-3664.

[119] E. De Cristofaro and R. Di Pietro. Adversaries and countermeasures in privacy-enhanced urban sensing systems. *Systems Journal, IEEE*, 7(2):311–322, June 2013. ISSN 1932-8184.

[120] E. De Cristofaro and R. Di Pietro. Preserving query privacy in urban sensing systems. In *Proceedings of the 13th International Conference on Distributed Computing and Networking*, ICDCN'12, pages 218–233, Berlin, Heidelberg, 2012. Springer-Verlag.

[121] X. Chen, K. Makki, K. Yen, and N. Pissinou. Sensor network security: a survey. *Communications Surveys & Tutorials, IEEE*, 11(2):52–73, 2009.

[122] D. G. Padmavathi, M. Shanmugapriya, et al. A survey of attacks, security mechanisms and challenges in wireless sensor networks. *arXiv preprint arXiv:0909.0576*, 2009.

[123] T. Kavitha and D. Sridharan. Security vulnerabilities in wireless sensor networks: a survey. *Journal of information Assurance and Security*, 5(1):31–44, 2010.

[124] 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. ISSN 1570-8705. Privacy and Security in Wireless Sensor and Ad Hoc Networks.

[125] P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk. Enhancing source-location privacy in sensor network routing. In *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pages 599–608, June 2005.

[126] M. Shao, Y. Yang, S. Zhu, and G. Cao. Towards statistically strong source anonymity for sensor networks. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages –, April 2008.

[127] K. Mehta, D. Liu, and M. Wright. Location privacy in sensor networks against a global eavesdropper. In *Network Protocols, 2007. ICNP 2007. IEEE International Conference on*, pages 314–323, Oct 2007.

[128] Y. Jian, S. Chen, Z. Zhang, and L. Zhang. A novel scheme for protecting receiver's location privacy in wireless sensor networks. *Wireless Communications, IEEE Transactions on*, 7(10):3769–3779, October 2008. ISSN 1536-1276.

[129] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[130] S. Chessa, R. Di Pietro, and P. Maestrini. Dependable and secure data storage in wireless ad hoc networks: An assessment of ds 2. In *WONS*, pages 184–198, 2004.

[131] Q. Wang, K. Ren, S. Yu, and W. Lou. Dependable and secure sensor data storage with dynamic integrity assurance. *ACM Transactions on Sensor Networks (TOSN)*, 8(1):9, 2011.

[132] Y. Ren, V. Oleshchuk, and F. Y. Li. A distributed data storage and retrieval scheme in unattended wsns using homomorphic encryption and secret sharing. In *Wireless Days (WD), 2009 2nd IFIP*, pages 1–6. IEEE, 2009.

[133] Y. Ren, V. Oleshchuk, and F. Y. Li. A scheme for secure and reliable distributed data storage in unattended wsns. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–6. IEEE, 2010.

[134] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. In D. Coppersmith, editor, *Advances in Cryptology — CRYPT0' 95*, volume 963 of *Lecture Notes in Computer Science*, pages 339–352. Springer Berlin Heidelberg, 1995.

[135] G. Blakley. Safeguarding cryptographic keys. In *Proceedings of the 1979 AFIPS National Computer Conference*, pages 313–317, Monval, NJ, USA, 1979. AFIPS Press.

[136] C. Avin and G. Ercal. On the cover time and mixing time of random geometric graphs. *Theoretical Computer Science*, 380(1–2):2 – 22, 2007. ISSN 0304-3975. Automata, Languages and Programming.

[137] C. Bettstetter. On the minimum node degree and connectivity of a wireless multihop network. In *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking &Amp; Computing*, MobiHoc '02, pages 80–91, New York, NY, USA, 2002. ACM.

[138] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005.

[139] R. Arratia, L. Goldstein, and L. Gordon. Poisson approximation and the chen-stein method. *Statistical Science*, 5(4):403–424, 11 1990.

[140] D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov chains and mixing times*. American Mathematical Society, 2006.

[141] L. B. Koralov and Y. G. Sinai. *Theory of Probability and Random Processes*. Springer, 2007.

[142] C. Bettstetter, G. Resta, and P. Santi. The node distribution of the random waypoint mobility model for wireless ad hoc networks. *Mobile Computing, IEEE Transactions on*, 2(3):257–269, July 2003. ISSN 1536-1233.

[143] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948.

[144] C.-W. Chen and Y.-R. Tsai. Location privacy in unattended wireless sensor networks upon the requirement of data survivability. *Selected Areas in Communications, IEEE Journal on*, 29(7):1480–1490, August 2011. ISSN 0733-8716.

[145] C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou. Toward secure and dependable storage services in cloud computing. *IEEE TSC*, 5, 2012.

[146] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. Provable data possession at untrusted stores. *ACM CCS'07*, .

[147] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik. Scalable and efficient provable data possession. *SecureComm'08*, .

[148] R. Curtmola, O. Khan, R. Burns, and G. Ateniese. MR-PDP: Multiple-replica provable data possession. *ICDCS'08*.

[149] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia. Dynamic provable data possession. *ACM CCS'09*.

[150] Y. Deswarte, J. Quisquater, and A. Saidane. Remote integrity checking. *IICIS'03*.

[151] F. Sebe, A. Martinez-Balleste, Y. Deswarte, J. Domingo-Ferrer, and J. Quisquater. Time-bounded remote file integrity checking. Technical Report 04429, 2006.

[152] P. Golle, S. Jarecki, and I. Mironov. Cryptographic primitives enforcing communication and storage complexity. *Financial Cryptography*, 2002.

[153] T. S. J. Schwarz and E. L. Miller. Store, forget, and check: Using algebraic signatures to check remotely administered storage. *ICDCS'06*.

[154] A. Juels and B. S. Kaliski, Jr. Pors: proofs of retrievability for large files. *ACM CCS'07*.

[155] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song. Remote data checking using provable data possession. *ACM TISSEC*, 14, 2011.

[156] F. Sebé, J. Domingo-Ferrer, A. Martinez-Balleste, Y. Deswarte, and J.-J. Quisquater. Efficient remote data possession checking in critical information infrastructures. *TKDE*, 20, 2008.

[157] M. Naor and K. Nissim. Certificate revocation and certificate update. *IEEE JSAC*, 18, 2000.

[158] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S. S. Yau. Efficient provable data possession for hybrid clouds. *CCS'10*, .

[159] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S. S. Yau. Dynamic audit services for integrity verification of outsourced storages in clouds. *ACM SAC'11*, .

[160] J. Yang, H. Wang, J. Wang, C. Tan, and D. Yu. Provable data possession of resource-constrained mobile devices in cloud computing. *Journal of Networks*, 6, 2011.

[161] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou. Enabling public verifiability and data dynamics for storage security in cloud computing. *ESORICS'09*, .

[162] Z. Hao, S. Zhong, and N. Yu. A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability. *IEEE TKDE*, 23, 2011.

[163] C. Wang, Q. Wang, K. Ren, and W. Lou. Privacy-preserving public auditing for data storage security in cloud computing. *IEEE INFOCOM'10*, .

[164] C. Chen and R. Curtmola. Robust dynamic provable data possession. Technical report, 2012.

[165] J. Plank and L. Xu. Optimizing cauchy reed-solomon codes for fault-tolerant network storage applications. *IEEE NCA'06*.

[166] M. Castro and B. Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM TOCS*, 20, 2002.

[167] K. D. Bowers, M. van Dijk, A. Juels, A. Oprea, and R. L. Rivest. How to tell if your cloud files are vulnerable to drive crashes. *ACM CCS'11*.

[168] R. Bragantini, M. Conti, and R. Di Pietro. Security in outsourced storage: Efficiently checking integrity and service level agreement compliance. *IEEE TSP'10*, 2010.

[169] D. F. Galletta, R. M. Henry, S. McCoy, and P. Polak. Web site delays: How tolerant are users? *JAIS*, 5, 2004.

[170] F. F.-H. Nah. A study on tolerable waiting time: how long are web users willing to wait? *Behaviour & IT*, 2004.

[171] J. Postel. Internet control message protocol. RFC 792, 1981.

[172] R. Impagliazzo and V. Kabanets. Constructive proofs of concentration bounds. *LNCS*, 6302, 2010.