Attacker Modelling in Ubiquitous Computing Systems

Papini, Davide; Sharp, Robin; Jensen, Christian D.

Publication date:
2012

Document Version
Publisher's PDF, also known as Version of record

Attacker Modelling in Ubiquitous Computing Systems

Davide Papini

Kongens Lyngby 2012
IMM-PHD-2012-295

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-PHD: ISSN 0909-3192

A mio nonno,
Vittorio Pozzesi

Summary

Within the last five to ten years we have experienced an incredible growth of ubiquitous technologies which has allowed for improvements in several areas, including energy distribution and management, health care services, border surveillance, secure monitoring and management of buildings, localisation services and many others. These technologies can be classified under the name of ubiquitous systems.

The term *Ubiquitous System* dates back to 1991 when Mark Weiser at Xerox PARC Lab first referred to it in writing. He envisioned a future where computing technologies would have been melted in with our everyday life. This future is visible to everyone nowadays: terms like smartphone, cloud, sensor, network etc. are widely known and used in our everyday life.

But what about the security of such systems. Ubiquitous computing devices can be limited in terms of energy, computing power and memory. The implementation of cryptographic mechanisms that comes from classical communication systems could be too heavy for the resources of such devices, thus forcing the use of lighter security measures if any at all. The same goes for the implementation of security protocols. The protocols employed in classical communication systems were not designed for the ubiquitous environment, hence their security has to be proven again, leading to the definition of new protocols designed specifically to address new vulnerabilities introduced by the ubiquitous nature of the system.

Throughout the network security community this problem has been investigated for some time now and this has resulted in some lightweight cryptographic stan-

dards and protocols, as well as tools that make it possible for security properties of communication protocols which are typical of ubiquitous systems. However the abilities of the ubiquitous attacker remain somehow undefined and still under extensive investigation.

This Thesis explores the nature of the ubiquitous attacker with a focus on how she interacts with the physical world and it defines a model that captures the abilities of the attacker. Furthermore a quantitative implementation of the model is presented. This can be used by a security analyst as a supporting tool to analyse the security of an ubiquitous system and identify its weak parts. Most importantly this work is also useful for system designers, who wish to implement an effective secure solution while developing their system.

Resumé

Inden for de seneste fem til ti år har vi oplevet en utrolig vækst af allestedsnærværende teknologier, der har gjort forbedringer mulig på flere områder, blandt andet energidistribution og forvaltning, sundhedstjenester, grænseovervågning, sikker overvågning og styring af bygninger, lokaliserings tjenester og mange andre. Disse teknologier kan klassificeres under navnet allestedsnærværende systemer.

Udtrykket Allestedsnærværende System daterer tilbage til 1991, hvor Mark Weiser hos Xerox PARC Lab første gang omtalte det skriftligt. Han forestillede sig en fremtid, hvor computerteknologi ville være smeltet sammen med vores hverdag. Denne fremtid er synlig for alle i dag: begreber som smartphone, skyen, sensor, netværk osv. er almindeligt kendt og anvendt i vores hverdag.

Men hvad med sikkerheden af sådanne systemer. Allestedsnærværende enheder kan være begrænset af energi, computerkraft og hukommelse. Brugen af kryptografiske mekanismer, der kommer fra klassiske kommunikationssystemer, kan være for tung for sådant udstyr, og dermed tvinges man til at anvende simplere sikkerhedsforanstaltninger, hvis nogen overhovedet. Det samme gælder for gennemførelsen af sikkerhedsprotokoller. Sikkerhedsprotokoller fra klassiske kommunikationssystemer er ikke designet til det allestedsnærværende miljø, og derfor skal deres sikkerhed bevises igen, hvilket fører til definitioner af nye protokoller der er designet specielt til at modvirke nye sårbarheder indført ved den allestedsnærværende karakter af systemet.

Dette problem er blevet undersøgt i netværkssikkerhedskredse i et stykke tid nu, og har resulteret i nogle letvægts kryptografiske standarder og protokoller, samt værktøjer der gør det muligt at kontrollere sikkerhedsegenskaberne af kommu-

nikationsprotokoller, der er typiske for allestedsnærværende systemer. Men de evner den allestedsnærværende angriber besidder er stadig udefineret og under omfattende undersøgelse.

I denne afhandling udforskes karakteren af den allestedsnærværende angriber med fokus på hvordan hun interagerer med den fysiske verdens og der defineres en model, der inkluderer angriberens evner. Endvidere presenteres en kvantitativ implementering af denne model. Denne kan bruges af en sikkerhedsanalytiker som et støttende redskab til at analysere sikkerheden af et allestedsnærværende system og identificere dets svage dele. Vigtigst, så er dette arbejde nyttigt for system designere, der ønsker at implementere en effektiv sikker løsning, under udviklingen af deres system.

Preface

This thesis was prepared at Informatics Mathematical Modelling, the Technical University of Denmark in partial fulfillment of the requirements for acquiring the Ph.D. degree in engineering. The Ph.D study has been carried out at DTU-Informatics under the supervision of Robin Sharp and the co-supervision of Christian Damsgaard Jensen (DTU-Informatics) in the period of three years between October 2009 and December 2012, the project was funded by the ITMAN Graduate School of DTU-Informatics.

The thesis introduces a new quantitative attacker model for ubiquitous computing systems. The model can be used as a powerful tool to assess the security of an ubiquitous system against the attacker capabilities.

The thesis consists of a summary report, seven chapters and three appendices. One of the appendices contains a collection of abstracts of the research papers written during the period 2009-2012, and published elsewhere.

Davide Papini

Lyngby, December 2012

Acknowledgements

Looking back at the last five years that I have spent in Denmark, and specifically the last three, the years of my PhD, there are many people that I would like to thank, and unfortunately they cannot all be mentioned here. My first thanks goes to my supervisor, Robin Sharp, with whom I share both scientific and musical interests; he followed me throughout all these five years, first as a professor, secondly as supervisor of my master project and finally as supervisor for my PhD project. His insights and his guidance was of great help both for my personal education and the research I carried out during my PhD studies. I would like to thank also Christian Jensen, for some very stimulating talks we had in the early stages of my research. A special mention goes to my office mates, Luke, Mads and especially Naveed with whom I frequently had stimulating discussions on different topics regarding research and security. A special thank goes to Nicola, who made me expand my research efforts, and to Alberto who contributed to a positive working environment.

I thank all the PhDs at ESE, IMM and passing by, with whom I shared the last three years, especially Massimo, Pietro, Laura, Fontas, Alessio, Domitian, Letizia, Roberto and all the others.

I thank my parents who strongly supported me in my choices, my grandparents and my brothers, especially Emanuele, always ready to spare some time for me. At last I thank my wife for her loving support and the patience she had during the writing of this thesis.

This thesis is dedicated to my grandfather, Vittorio, who had a profound impact on the person I am today.

Contents

List of Figures

List of Tables

List of Listings

List of Acronyms

ATK Attacker.

BAN Body Area Network.

BCC Body Coupled Communication.

CDF Cumulative Density Function.

CPA Correlation Power Attack.

CRC Cyclic Redundancy Check.

CTMC Continuous Time Markov Chain.

DPA Differential Power Analysis.

DTMC Discrete Time Markov Chain.

EH-WSN Energy Harvesting Wireless Sensor Network.

ENISA European Network and Information Security Agency.

IDS Intrusion Detection System.

IM Intervention Module.

IPS Intrusion Prevention System.

ISO International Organization for Standardization.

JTAG Joint Test Action Group.

LAN Local Area Network.

MAC Media Access Control.

MAN Metropolitan Area Network.

MDP Markov Decision Process.

NFC Near Field Communication.

NIST National Institute of Standards and Technologies.

OSI Open System Interconnection.

PAN Personal Area Network.

PC Personal Computer.

PDF Probability Density Function.

PTA Probabilistic Timed Automata.

QoS Quality of Service.

SPA Static Power Analysis.

UC Ubiquitous Computing.

WAN Wide Area Network.

WBAN Wireless Body Area Network.

WLAN Wireless Local Area Network.

WSN Wireless Sensor Network.

Chapter 1

Introduction

It was the year 1991 when the term *Ubiquitous Computing* first appeared in the literature.

> "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it."
>
> *Mark Weiser*, [63]

The author foresaw the evolution from the PC seen as a black-box to a reality where computers are fully integrated with the environment. Following his work in [64] and [65] Weiser combines ubiquitous computing with the concept of *invisible computing*. The concept expresses the fact that we humans use technologies which we are not aware of or that we do not directly perceive while we use them. To make an everyday example: one of the biggest revolutions in the past century, the 20th, was brought by the invention of television. While we watch it, we do not think that images are carried through radio waves and translated into visual stimuli by an electron beam (in old televisions) or by the stimulation of a transistor which generates the correct combination of photons (in the most modern apparatus). The same goes with printing technologies, automotive and similar. The idea here is that we are not aware of the number of devices nor

the type or the technologies involved in the process, simply because they have become an active part of our everyday life.

Ubiquitous Computing (UC) is a term that defines the third era of modern computing (cf. [41]). In the first era, many people were using single room sized computers, called mainframes; this happened because of the costs of technology and for its dimensions. The second era came with the invention of the Personal Computer (PC), a desk-size device meant for one person. The third era, ubiquitous computing, is the one that we now live in: we use multiple devices, PCs, tablets, smartphones etc. and we are barely aware of the underlying technology that interconnects them. The thing that differentiates these three eras is the number of devices. Figure 1.1 shows a conceptual graph of the evolution of the three eras with respect to the number of devices (source [41]).

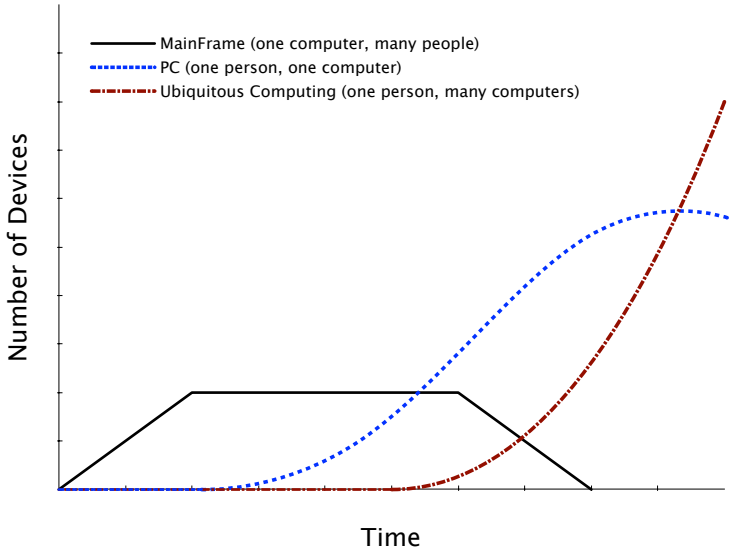


Figure 1.1: The Three Eras of Computing - Source [41]

In the early stages of his research, Weiser developed the first ubiquitous system, the *ParcTab Ubiquitous System* [62]. Together with his research team, he defined the dimensions of devices composing it being inspired by the units of length *inch*, *foot* and *yard*. An inch-sized device is a device that easily fits into a pocket and that can be forgotten around (very much like a smartphone today), a foot-scale device is something bigger that can still be carried around (e.g. modern tablets) while a yard scale device is something that cannot be moved (a television for example). The *ParcTab Ubiquitous System* was composed of three interface devices: the inch size ParcTab, the foot size ParcPad and the yard size

Liveboard. These devices were interconnected through an InfraRed wireless network and they could access any kind of resources or applications: mail, shell, calendars, memo, documents, control of heating and air-conditioning etc. Furthermore the *ParcTab Ubiquitous System* had location protocols in place so that it was possible to locate every device within the network, along with the person carrying it.

Ubiquitous computing has evolved from Weiser's work and nowadays it comprises several technologies like pervasive computing, ambient intelligence, augmented reality, physical computing, internet of things and many more. Such technologies are used for a variety of applications, including:

- Health care, to remotely monitor the status of patients, to run a therapy, to give access to patients' data whenever needed.
- Smart homes, home environments which react to the activity of the people moving and living in them.
- Building sector, to monitor the integrity of infrastructures such as bridges or buildings.
- Energy, to regulate the energy distribution throughout an energy distribution network to minimise energy losses.
- Border surveillance, to monitor trespasses over a monitored area. [59].
- Military, for real time monitoring of battlefields and soldiers conditions, both psychological and clinical [20].

The systems based on ubiquitous technologies are composed of small and diverse devices that can be limited in terms of energy, computing power and memory. Furthermore, due to the technological evolution we have witnessed in the last decade, these devices are usually fully interconnected through wireless/radio networks that possibly span large uncontrolled areas. Figure 1.2 shows an overview of the current and most common wireless communication technologies, these are classified by data rate, range of communications and mobility of the devices.

As wireless technology has become more and more accessible to everyone, the need for security has become a primary issue. In addition cybercrime activity has increased significantly within the last five years, while international and national institutions has recently started to address the problem by issuing guidelines and best practices for corporations working in the public sector or for government infrastructures. The UK was one of the first countries that

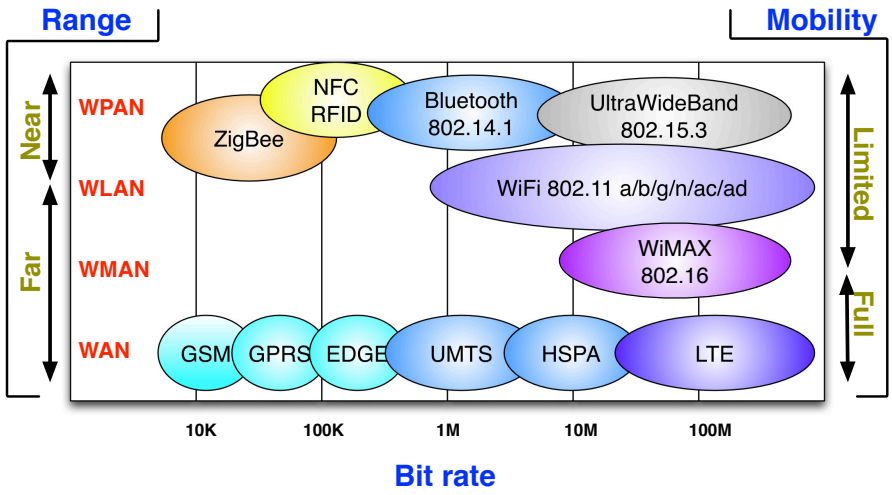


Figure 1.2: Wireless Communication Technologies Overview

issued official guidelines to address cyber security for the private sector [25]. According to a report published by the European Network and Information Security Agency (ENISA) [44], recently cyber criminals have become interested in mobile devices and ubiquitous environment, hence the issue of securing these devices has become even more pressing.

1.1 Motivation of this Work

When it comes to Ubiquitous Computing, implementing security is not an easy task, for each property we add (e.g. integrity, authentication etc..) the overhead, in terms of computing power and energy used to transmit supplementary bits, can significantly decrease the energy of a device, thus making systems useless in practice. For this very reason there has been extensive research to develop new cryptographic algorithms, which are lightweight as regards the above mentioned considerations; this led to new standards such as [36].

Furthermore there are applications where the security level is not required to be the highest and this level may depend on the context the system is operating in. In [60] for example the authors define a Wireless Sensor Network (WSN) messaging system where the security level depends on the available energy. When it comes to defining such systems, where the security conditions can be relaxed, a

natural question arises: "What is the acceptable security level?". This question can be seen as the synthesis of a series of other questions like:

- What is it that needs to be protected/secured?
- What are the losses if security is compromised?
- What is the outcome in terms of overall security if one or multiple parts of the system are independently compromised?

In order to answer these questions and ultimately support the development of ubiquitous systems under the restrictions mentioned earlier, the need for a study and a characterisation of the attacker capabilities arises.

1.2 The Attacker in the Ubiquitous Environment

As ubiquitous systems are highly integrated in the environment surrounding them, their behaviour depends on the context they are immersed in. The attacker can therefore influence a system on multiple front lines, those pertaining to the cyber world, which mainly characterise classical approaches to attacker modelling, and those involving the interaction of the attacker with one or multiple devices from a physical perspective. The latter is not an entirely new concept, social engineering for example is a similar one: here the attacker gets information through her social interactions, hence she goes beyond the cyber world, getting to the real one and exploiting her social dimension.

Another aspect that needs to be taken into consideration is that the attacker can attack a system both at a protocol and at a cryptographic level. This means that in addition to exploiting protocol weaknesses, the attacker may have the ability of breaking the encryption of a system by exploiting vulnerabilities in the implementation or in the encryption algorithm itself. The WEP (Wireless Equivalent Privacy) case is a well known example of an encryption algorithm broken in numerous ways (for example see [61]). Other well known encryption standards show some weaknesses, the AES itself seems vulnerable for example to a class of attacks called *related keys* attacks [16], which have a reduced time and memory complexity with respect to the naive brute force attack. The cipher is still secure in terms of practicality of the attack, the National Institute of Standards and Technologies still approves of its use [9]; nevertheless this example shows that the even the strongest cipher may be compromised by a cryptographic attacker to some extent.

With respect to her physical dimension the attacker may tamper with devices and access their memory, thus getting hold of data as well as cryptographic keys or identities. In the case of energy constrained devices, the attacker may also reduce their energy or control it by increasing or decreasing the amount of energy or by switching the device off and on. The attacker may as well reprogram a device to change its behaviour for a specific purpose or inject a malicious device into the system.

1.3 Scope of the Thesis and Contributions

The scope of this thesis is to define an attacker model which takes into account both the physical and cyber aspects. Furthermore, as ubiquitous computing systems may be subject to different levels of security, the security analyst has to be able to perform a quantitative analysis of a system; part of this thesis is devoted to researching a method to quantitatively assess the security of a system.

The approach taken in this thesis is that the attacker is defined by three properties: interventions, presence and time, which we call dimensions. These dimensions define what the attacker can do, where she does it and for how long. Furthermore basic actions the attacker may carry out are defined. These actions classify what are the attacker abilities with respect to the communication channel, the physical interaction with devices and finally with respect to cryptography. In the perspective of a quantitative model, the strength of the attacker is also discussed; this is important as systems requirements may differ depending on the application. Military applications for example should be designed to face strong and motivated attackers, while an environment monitoring systems may not face similar attackers ever in their lifetime.

In addition to the model definition, a quantitative probabilistic model is defined and implemented. The purpose of this is to produce a tool that may be used as a support to perform a security analysis of a system. It is important to say that the experience of a security analyst is still needed, the quantitative model is intended only as a support for the analysis.

Throughout this thesis arises the problem of modelling the ability of the attacker to eavesdrop a communication from a wireless device. To deal with this problem, a probabilistic model is implemented, this can be used to compute the probability of eavesdropping a transmission. The model accounts for the mobility of the parties involved, which can be either deterministic or probabilistic, the nature of the transmission technology, and the geometry of the space the

parties move in.

1.4 Related Work

In this section relevant related work is mentioned. As in the case of attacker models in classical communication systems, there is a large body of work (here with respect to ubiquitous systems), so the presentation here will only discuss the most significant ones. In addition in the literature it is possible to find some documents which address some of the physical features of the attacker.

Classical attacker models often assume cryptography as a secure black box and model attackers as entities which control the network. The Dolev-Yao attacker (defined in [29], 1983) is a well known example of a powerful network attacker: she can control the network and hence intercept, replay or destroy messages; this attacker can also perform encryption and decryption, but she can get hold of cryptographic keys only by learning them from a third party. It is probably the best known attacker model in the network security community.

In the literature it is also possible to find other models that provide the attacker with some cryptographic ability. The Bellare-Rogaway model [13] augments the attacker with the ability of initiating new authentication sessions and to discover keys in a probabilistic fashion (i.e. coin toss).

The Rubin-Shoup model [55] extends the Bellare-Rogaway model to deal with the introduction of a new technology, the Smart Card, by defining probabilistic oracles that can be queried by the attacker. This extension was needed as the SmartCard has no timing functionality hence it cannot verify the freshness of a message. This is probably the first example where the physical dimension of the attacker begins to emerge, as the idea here is that attacker can query the smart card separately and independently from the system for instance by stealing it for a short time.

In [23] Creese et al. approach the problem of whether the Dolev-Yao attacker is still appropriate for ubiquitous systems. They conclude that the ubiquitous nature of the system may need to assume the existence of multiple communication channels which are subject to different threat models. In their analysis they also specify that the attacker may be restricted in her actions, specifically it is unlikely that the attacker may perform multiple actions that may interfere with each other, at the same time: they consider the example where the attacker may overhear messages or be able to destroy them by some sort of jamming, but not do both things at the same time. In [22] the same authors refine their

threat model to perform a security protocol analysis which takes into account the existence of multiple channel in the presence of restricted attackers.

Basin et al. in [10] and afterwards in [11] present a formal model for modelling and reasoning about "physical security protocols". These are protocols that exploit physical features to securely perform tasks such as localisation and time synchronisation. Specifically the authors formalise physical properties such as communication, location and time, taking into account for example how much time a message takes to be dispatched between nodes given their distance and the communication medium. This model can be used to verify security properties of protocols, but in terms of cryptographic abilities it has the same restrictions as the Dolev-Yao model.

Furthermore again Basin et al. in [12] discuss the problem of devices which have been compromised by an attacker. Specifically they deal with the leakage of cryptographic material (e.g. long term keys) that the attacker may use to get access to confidential data. In their work they developed a framework that keeps track of compromised data, so that it is possible to assess to which level the security of a system is threatened. This is still done at a protocol level, i.e. knowing some cryptographic keys and having compromised some nodes, what are the information that the attacker could get from a protocol execution.

An idea worth mentioning which can be used to understand attacker capabilities is attack trees [54]. These are representations of attacks as a tree, the root node is the attacker goal while the child nodes represent attacker sub-goals, needed to achieve the root goal. Kordy et al. in [39] and [40] expand attack trees to include defence strategies as well. This makes representation of attacks more effective as defences are represented as well, together with attacker abilities that aim to defeat them.

To the knowledge of the author there is no quantitative model that addresses the attacker, as it has been defined within this work, in its entirety. This makes this work even more important and the motivation for it stronger.

1.5 Structure of the Thesis

The thesis is divided into seven chapters, an introduction (this chapter), four core chapters, a chapter summarising results and containing final remarks, the conclusions and three appendixes. At the end of each chapter there is a brief summary of the contents of the chapter. The thesis is organised as follows:

- Chapter 2 defines the attacker model by:
  1. Introducing the modelling methodology.
  2. Identifying the attacker dimensions.
  3. Defining the relationship between them.

  At the end of the chapter two case studies are also proposed.
- Chapter 3 addresses the problem of the quantitative implementation of the model by:
  1. Finding the appropriate modelling language.
  2. Explaining how it works and how a model can be checked with it.
  3. Explaining how the model was implemented.
- Chapter 4 addresses the problem of defining the probabilities for the attacker actions. Within this chapter the aforementioned eavesdropping model is defined and implemented.
- Chapter 5 shows the quantitative analysis of the two proposed case studies.
- Chapter 6 discusses the results and illustrate how the model can be used by security analysts and system designers. In addition it compares the thesis to the related work.
- Finally Chapter 7 draws the conclusions and discusses future work.

The appendixes include a small specification of the modelling language, specifications about the eavesdropping model and the abstract of papers published by the author.

1.6 Conventions and Intended Audience

Throughout this manuscript several conventions are used, to make the process of reading more fluent and the contents more understandable:

- Literal conventions:
  - The attacker is always referred to with a female gender (e.g. she, her etc.).
  - The analyst is always referred to with a male gender (e.g. he, him etc.).
  - The author of this thesis is also referred to as the writer. The writer may obviously play the role of the analyst when required.

- Notations:
  - $P(A)$ refers to the mathematical probability of event A happening.
  - $P(A|B)$ refers to the conditional probability of A happening given that B already happened.
  - $A \rightarrow B$ refers to the transition from state A to state B.
  - $A \nrightarrow B$ means that it does not exist a transition from state A to state B.

As for the audience, the reader is expected to have a general knowledge about network security and security in general. Particularly he should have a general knowledge of cryptography, protocol theory, and the OSI model in general [34]. The reader should have a good knowledge of probability theory and a basic knowledge of Markov models. Finally this thesis, specifically Chapter 2, can be read also by people who do not have any knowledge about security and may want to know about threats in ubiquitous systems as they are involved in the design or implementation process of them.

Chapter 2

The Attacker Model

This chapter defines the attacker model. The first section describes the methodology by addressing two questions: is it possible to define a comprehensive attacker model for such a plethora of systems associated with Ubiquitous Computing? Moreover, does it make sense to do this? Section 2 defines the attacker model and Section 3 presents an initial qualitative analysis of two cases of study.

Part of the work described in this chapter has already been published in [26], [28] and [27].

2.1 Defining a Methodology

Before going into the methodology, a simple idea may help to explain the methodology: Historically security was born to protect communication that needed to be private and confidential. The content of this communication ranges from innocent private messages to military orders, political strategies etc., information whose disclosure would have changed the balance of power between opposite forces. Methods of concealing messages went from folding the message and hiding it in secret places, to letter substitution performed according to a secret scheme (e.g. classic alphabetic and polyalphabetic ciphers like the Caesar

Cipher). The attacker then tried to unveil messages and, when this succeeded, new and more secure systems were defined in order to address the new threat.

Figure 2.1 illustrates this *Attacker System Causality Loop* where current attacker capabilities evolve and new ones emerge in each cycle. This dynamics can be related to technology development or simply to the attacker being smarter. This results in the evolution of the systems to address the newly discovered security issues. If we call the set of abilities/capabilities of the attacker an *Attacker Model*, it should be clear how the design of a system and its security is closely linked to the *Attacker Model* and vice versa. Indeed security goals are also defined regardless of the attacker's existence (e.g. the content of a letter should be known only to the intended receivers or the illness affecting a patient should be known only to the doctors and relevant medical staff), but in this work we are only interested in the $SystemDesign \leftrightarrow AttackerModel$ relationship.

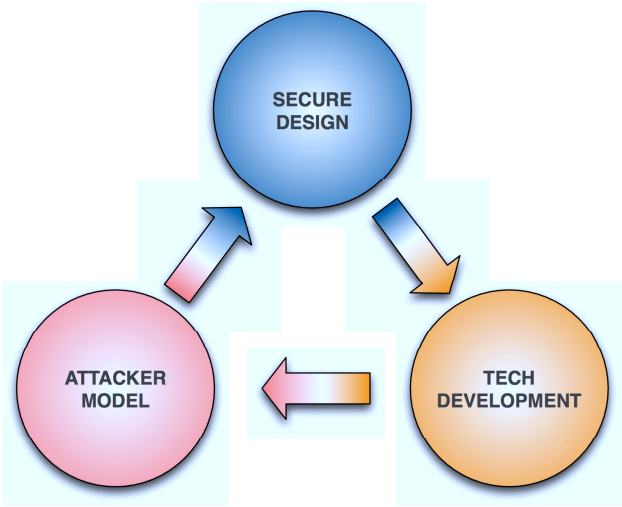


Figure 2.1: Attacker System Causality Loop

To put it into the contest, the writer would like to give an example of three runs of the loop within recent history of security.

Loop One - In 1983 Dolev and Yao wrote a well known paper[29] where they formally defined an attacker model, the "Network Attacker" (described informally in [48]), where an omnipotent attacker controls the network over which messages are exchanged. This attacker does not have any power over the cryptographic functions used by the communication processes, apart being able

to perform encryption/decryption given that the key is known.

This model is a milestone in the history of security, nevertheless it relies on very strong assumptions, among which:

- The attacker does not have any cryptoanalytical capability.
- The attacker does not have any "physical" dimension.

Loop Two - Starting from the first assumption, the Dolev-Yao model does not deal with an attacker having cryptoanalytical skills, which would allow her to find e.g. specific cypher-plaintext pairs or encryption/decryption keys. In 1993 Bellare and Rogaway in [13], defined a model based on the random oracle paradigm, which connects cryptographic theory to cryptographic practice. This model depicts a scenario where the attacker is able not only to control the medium, but also to initiate new authentication sessions. Here participants are represented as oracles which can be run by the attacker. Within this model it is possible for an attacker to "open" sessions (discover keys) in a probabilistic fashion (i.e. coin toss), and a protocol is said to be secure as long as the attacker is not able to create or modify a message that would be afterwards authenticated by the system.

Loop Three - In the 90s and more prominently in the 2000s, a new device made it into the world as "secure repository" of cryptographic material: the Smart Card [35]. Although several patents about SmartCards appeared in the late 70s, this technology became extensively used with mobile phones in the 90s and with payment systems (e.g. EMV [31]) between the late 90s and early 2000s. The security of such devices has been very much discussed both in terms of physical and protocol security. As a matter of fact the mere inclusion of the smart card into the system has produced in some cases severe security breaches due to the nature of the smart card device (see for example [46]). In 1996 Rubin and Shoup in [55] extended the Bellare-Rogaway model, and enabled it to deal with the use of Smart Cards. Specifically they modelled Smart Cards as probabilistic oracles and, within the model, they defined also the possibility for the attacker to know messages exchanged between smart card and a communication process at any time. The main difference between smart cards (i.e. Rubin-Shoup model) and computer based (i.e. Bellare-Rogaway model) key exchange mechanisms, is that a smart card cannot keep track of time, and it always acts as independent identity.

2.1.1 Problem: How to Define a Universal Model for Such Diverse Systems?

As the field of Ubiquitous Computing is enormous and its applications virtually infinite, attacker modelling would seem rather an impossible task at first glance. This is mainly due to the fact that the technologies involved are numerous (too many protocols and cryptographic primitives), and even worse they increase every year both with standard and proprietary solutions; the task of modelling a general system itself could be unfeasible. Moreover even if a general model could be produced, the chances are its applicability to real cases would be of no use due to its generality.

Thinking about this complexity and variety, when defining an attacker model for such systems one needs to take into account the following:

- The space of variables and features that come out has to be kept to a minimum as it could easily explode.
- The resulting model, although very detailed and precise, could be impossible to implement and/or analyse.

Therefore the need for a strong and sound methodology which addresses the above issues step by step arises.

2.1.2 Solution: Inductive Approach

As security is ultimately about taking theory into practice, and attacking is mostly about practice, a practical approach seems the most suitable one. In order to identify the attacker the writer decided to use an Inductive approach: first an attacker model for a subset of Ubiquitous Systems is defined, possibly the ones that are more critical or where deployment of security is somehow troublesome; then the model is expanded and tailored according to the system one wants to analyse.

Defining the attacker, in practice, translates into answering questions like:

- What are the attacker's capabilities, i.e. what is an attacker able to do?
- What is the attacker environment, i.e. where is she located?
- How much time does she have to perform an attack?
- What goals does the attacker have?

- How skilled is the attacker?
- How motivated is the attacker for performing her actions?

Moreover considering how systems are designed helps in understanding attacker strategies and give us a clearer picture of the attacker (remember the concept expressed in Figure 2.1).

Looking at current trends and emerging technologies, and after considering what are the systems that are more critical in terms of security, the writer decided to use three types of systems as a basis for the inductive approach:

- Body Area Networks for healthcare applications.
- Legacy Wireless Sensor Networks.
- Energy Harvesting Wireless Sensor Networks.

Hence from the analysis of these systems, an attacker model is defined. This can be then extended to ubiquitous systems in general.

These systems are a good and representative starting point, as they have strong similarities as well as differences:

- Similarities
  - They are composed of individual nodes communicating through radio waves.
  - Most devices are constrained in terms of memory and computing power.
  - Both systems are designed to monitor an object (e.g. environment for WSN and vital signs for BAN) and perform actions based on collected data.
- Differences
  - Energy:
    - WSNs have more stringent energy constraints.
    - For BANs energy is not that critical since devices can often be easily recharged.
  - Space:
    - WSNs are distributed over a larger area than BAN, from $\simeq 50m$ up to several kilometres.
    - When WSNs are scattered over a large area, nodes usually do not get recovered (mainly for economical reasons).
    - BANs are located on the person and their application focus on the body and on the immediate surrounding environment.

- Duration:
  - WSNs are designed to last for years, EH-WSNs can last virtually indefinitely.
  - BAN nodes' energy lasts for days (sometimes even less than a day).
- Technology:
  - Opposite to BAN, WSNs have usually a more complex Data link and Network layer[1] as they need for example to implement routing algorithms and they often use non-standard solutions.
  - BANs make wide use of standard technologies which are more common (e.g. Bluetooth and recently NFC).

2.2 Identifying Cyber Physical Attacker

The attacker domain has changed radically since computer science and computer security have been born. After reading Chapter 1 and the first part of this Chapter, it should be clear to the reader that attackers have a physical dimension as well as a cyber one. But what does this mean practically speaking? How do we identify and define this physical dimension?

In the following part of this section we are going to discuss it by defining three dimensions: *Intervention, Presence* and *Time* (a similar approach has been taken by Benenson et al. in [15] and [14]). Section 2.2.1 defines Presence and Time while Section 2.2.2 Defines Intervention by first identifying the atomic actions the attacker can make.

The writer afterwards will explain how the three dimensions interact with each other and how are they to be interpreted with respect to attacker strength.

2.2.1 Presence and Time

Presence and Time are two dimensions which identify the attacker in space and time. As the attacker is cyber-physical, these two dimensions have a physical and a cyber connotation as well. These are closely linked as a physical feature can be directly related to a cyber one, moreover a physical feature can enable a cyber one and vice versa. Physically tampering with a device for example enables the cyber control of it, while the remote control of say a smoke detector

[1] As in the OSI model in [34].

could raise a fire alarm, as the attacker was there herself pushing the alarm button. Therefore, while considering these dimensions, one has to take into account this cyber-physical aspect.

Presence - Defines to what extent and over which area the attacker exerts some influence or control. As we have already seen in Chapter 1, Ubiquitous Systems can be schematised as a set of nodes interconnected by a network; within this schema presence is defined to be either *local*, *distributed* or *global*. Figure 2.2 gives a graphical representation of presence.

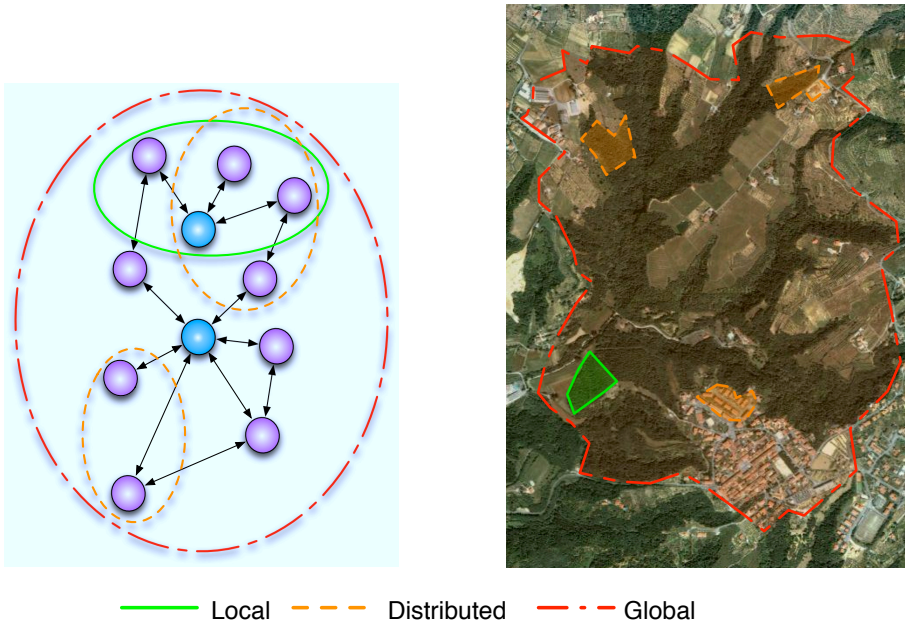


Figure 2.2: Presence

Presence is strongly dependent on the system at hand, more significantly than other dimensions. It is not easy to differentiate between local, distributed or global attacker in a Body Area Network, where all the devices are separated by tens of centimetres. This is clearly not the case for wireless sensor networks, where the typical device-to-device distance goes from meters to hundreds of meters, so identifying presence is more straightforward. Moreover in Body Area Networks, looking at the fact that every device performs a specialised operation (e.g. a temperature sensor, a heart meter, glucose meter or an insulin pump), compromising one sensor could very well mean that the whole system is compromised, thus getting a localised action to produce a global effect. On the other

hand, in systems such as Wireless Sensor Networks, there are more devices in terms of number, and more of a kind (e.g. more than one temperature, humidity or pressure sensor, more than one routing node, more than one processing node etc.), therefore the compromising of one does not imply the compromising of the whole; moreover security is usually dealt with by a distributed dynamic approach (e.g. key distribution algorithms based on master keys and peer-to-peer session keys). The main difference between WSNs and BANs is that the latter are seen as a whole system, which needs all its parts to function properly, while in the former each device can be seen as independent, and its existence is not critical to the correct operation of the network.

Time - Complementing *Presence*, *Time* expresses the time span which the attacker needs or uses or has at her disposal to perform an attack. It is not merely a problem of defining the passing of time strictly in terms of seconds, minutes or hours (seconds for a system could mean hours or days for another e.g. finding a 40-bit key takes much less time than a 128-bit one). In this work *Time* is generically identified as time-frames that are labelled either as *short-span*, *long-span* or *multiple-spans* (see Figure 2.3). The first two relate to a continuous time interval that can be either "long" or "short". Multiple-spans refers to the case where an attack needs multiple related intervals to be carried out (e.g. replay attacks).

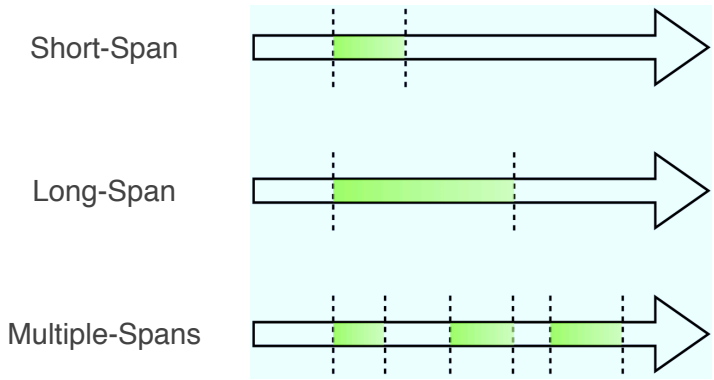


Figure 2.3: Time

In some cases multiple-spans can also be a combination of short-spans and long-spans. To give a simple example think about a replay-attacks: it happens over a multiple-span, a combination of eavesdropping and transmitting actions which can be regarded as short-spans. One of the case studies in Section 2.3 will give a more detailed and practical example of this.

2.2.2 Atomic Actions and Interventions

At this point *Intervention* is the only dimension that remains to define. It is the key dimension of the model as it describes the actions an attacker can perform.

In order to better understand intervention and relate it to specific systems, the writer identified a set of *Atomic Actions*. These are the basic bricks of intervention. Depending on the system at hand, one can derive the forms of intervention in a relatively straightforward manner by identifying the atomic actions that apply to it through the analysis of their feasibility.

Atomic actions fall into into three sets:

- Medium/Channel.
- Physical.
- Cryptography.

Medium/Channel atomic actions are those that the attacker performs on the communication channel or the medium (e.g. the electromagnetic domain for radio based communication systems). Some of them are typical of the *Network Attacker* (as in [29]), they are actions that solely interact with messages and/or data sent over the channel: `listen`, `inject`, `intercept`, `destroy`, `modify`. The other two atomic actions of this set are: `localise`, `selective block of destination/source communication links`. The first one is the action of localising a device. It is different from system to system and implies a combination of `listen` and some knowledge about the geometry of the environment. The second one is very interesting, it refers to the ability of blocking the communication in one way only, whenever the attacker needs it. This can be very powerful in presence of asynchronous communications.

Physical atomic actions aim to define the actions the attacker can perform in the physical world. These are: `tamper`, `switch on/off` (e.g. by removing and inserting battery), `decrease energy`, `increase energy`, `crash device` and `forge sensor reading`. Indeed these actions are not exhaustive, `tamper` for example means that the attacker accesses the device hardware and tampers with it, but the outcome of this actions could be numerous. An attacker could tamper with the device to get access to its memory, to reprogram it or to perform some sort of cryptographic actions (e.g. side channel attacks). `Switch on/off`, `decrease energy` and `increase energy` concern with physical actions that influence the power and energy of the device. It is important to stress the "physical" component in this case, since the energy of a device could be decreased also by using *Medium/Channel* actions (e.g. inject traffic) to force devices to lose energy by transmitting; `decrease energy` instead has

to be physical (e.g. direct discharge of a battery, or interference with the energy harvesting unit). Finally `crash device` is the physical destruction of a device while `forge sensor reading` refers to the case where the attacker forges the reading of a sensor for example by warming up a temperature sensor or alarming a smoke detector.

Cryptography is a set of atomic actions concerning cryptographic abilities of the attacker. These are either:

- Mathematical: the attacker has cryptoanalytical abilities.
- Resource based: the attacker has the resources to perform heavy cryptographic operations like factorisation or brute forcing of long keys ([57] discusses the key length against attacker's resources).
- Technical: the attacker exploits hardware vulnerabilities or insecure implementations. [7] is a striking example, here the authors managed to bypass the physical security protection of the target device (a device used both for military and civilian application), and discover the access key and get full read/write access to the secure protected memory.

Cryptography atomic actions are summarised in `break encryption` (e.g. key attacks), `find key` (e.g. by SPA, DPA [37]), `exploit insecure crypto implementation` (weak random number generator or re-keying methods, side channel attacks).

The role of atomic actions is to give a better understanding of interventions and help to identify the abilities needed by the attacker to perform them, especially when it comes to apply them to very different systems.

Intervention - As already specified in Section 2.1.2, this work bases on BANs and EH-WSNs since these are more critical, as they have limitations which affect their secure design. Nevertheless the interventions specified here can be extended to other systems. There are nine forms of intervention:

i. *Destruction*: the attacker can destroy one or more devices[2].

ii. *Eavesdropping*: the attacker can receive and store messages sent between devices.

[2]Here the term device is generic for BAN and WSN where it refers to nodes or sensors.

iii. *Data Knowledge*: the attacker can acquire the data stored on one or more devices (e.g. by dumping the whole memory).

iv. *Disturb/Partial Data Modification*: the attacker can partially modify data on a device (without direct access to the memory e.g. by injecting data into the network).

v. *Full Data Modification*: the attacker can fully modify data stored on a device (with direct access to the memory).

vi. *Reprogramming*: the attacker can reprogram a device.

vii. *Device Injection*: the attacker inserts new devices into the network.

viii. *Energy Reduction*: the attacker can reduce the device energy, or control its depletion rate.

ix. *Energy Control*: the attacker can exploit the energy level of a device in a malicious way.

Some of them express rather simple actions (e.g. destruction), while others can be more complex depending on the system at hand. Interventions can be composed to achieve more complex and different purposes depending on the attacker will.

Destruction Is the most simple and basic intervention: the destruction of a device. It happens when a device ceases to exist within a network or a system. Although it seems a pretty physical intervention it is not; it could very well be that the device is believed to be destroyed by the system but in fact its connection has been severed somehow (e.g. by taking out the battery, depleting its energy, DoS attack, the node moved out of range). Anyway the outcome of this is that the very same device will never come back into the network or system.

Eavesdropping Is the act of listening to the communication channel. It is the only passive intervention: the attacker does not alter the system in anyway if he just overhears communications. It could indeed happen that eavesdropping is the result of another active intervention, e.g. a node reprogrammed to relay messages to the attacker; nevertheless this intervention reflects the fact the attacker gets hold of a packet. Furthermore eavesdropping a packet does not mean that its contents are intelligible: it could very well be that messages are encrypted and the attacker does not have the key nor the abilities to decrypt it. Nevertheless the attacker could be able to get other information out of the messages such as source and destination, type of message, traffic characteristics. Starting from that,

using statistical methods such as traffic analysis, she could recover more detailed information about e.g. network topology or the role of specific devices. Eavesdropping is also the first step in several attacks as it is used to probe networks for devices and physically locate them.

Data Knowledge Expresses the ability of knowing part or even the whole content of the memory of a device. This can be done either by dumping the memory of the device or by extracting the value from the environment. The contents of the memory of a device depend on its activities and on the environment it is surrounded by. Now, depending on the value the attacker wishes to know, in some cases this could be evaluated from the environment (e.g. the temperature value stored in a sensor). It could also be that the attacker gets hold of cryptographic material (e.g. keys, nonces, identities, etc.), thus giving her legitimate access to the system.

Disturb/Partial Data Modification Is the act of modifying part of the data stored in the device memory by disturbing the legitimate functioning of it. This is done without direct access to the memory of the device. The value is fed into the memory by changing the corresponding logic or physical value. In the case of a temperature sensor for example, a disturbing action would be to change the local temperature of the sensor thus producing a false information. To give an example regarding logic layers, an attacker could tamper with routing tables or exploit MAC protocols by injecting false data into the network.

Full Data Modification Is the ability to completely modify the data memory[3] of a device. Contrary to *Disturb/Partial Data Modification*, this intervention assumes that the attacker has physical access to the device. Needless to say that this is a very powerful intervention: modifying the whole memory of a device could radically change its behaviour. Thinking about security only, a device could be forced into accepting another one as legitimate, and therefore exchange confidential data with it.

Reprogramming Is by far the most powerful, intrusive and possibly dangerous intervention: the reprogramming of a device. When an attacker is able to carry it out, she is in complete control of a legitimate device and the actions she can take are limited only by her own imagination apart from the device specifications of course. Reprogramming could be done locally, with direct access to the device, but it could very well be the case that legitimate remote mechanisms are exploited (e.g. firmware upgrades as in [58]). Surely reprogramming can be also used to achieve other interventions, but this will be explained better in Section 2.2.3.

[3]This is the memory that contains the data. The memory containing the program is assumed to be a different one.

Device Injection Is the injection of new devices in the network or the system in general. This means that the attacker is able to have the system accept a new device, with all its implications: the device is authenticated and trusted, a legitimate part of the system. This intervention is as powerful as *Reprogramming*, since an injected device's behaviour depends on its programming, which in this case is decided by the attacker; but there is more to it. A reprogrammed device still has the same hardware as a legitimate one. In the case of an injected device this is not true anymore. The attacker could choose a device which has a different hardware and therefore different capabilities. Even just a more capable processing unit or more memory could give the attacker a substantial advantage, especially in systems where devices are limited by design. Like *Reprogramming*, *Device Injection* could very well be a step to carry out more complex attacks; again we will see more in Section 2.2.3.

Energy Reduction With this intervention the attacker aims to reduce the energy of a device. This reduction can be partial or complete, depending on the attacker goals and on the system resilience. The reduction can be the result of direct (i.e. on-device) or indirect approaches. In the first case the attacker would need to have physical access to the device whilst in the latter she would reduce energy by stimulating device activity (e.g. transmission, computing, actuators). This intervention together with *Energy Control*, comes prominently into the picture with systems composed by energy constrained devices, and more generally with systems whose behaviour strongly depend on the energy available. This is the case of WSNs and EH-WSNs where nodes have limited energy which they need to measure out accurately, especially when it comes to security: the execution of cryptographic primitives has an energy overhead that can easily kill the energy supply of the device. The analysis of the energy cost of cryptography and communication has been researched throughout the last ten years. [51], [24] and [56] show that the cost of computation is often negligible with respect to the cost of transmitting and receiving data (i.e. the overhead due to the secure implementation).

Energy Control While energy reduction expresses the ability of reducing the energy of a device, *Energy Control* completes the scenario by adding the ability to control the amount of energy of the attacked device, that is not only its depletion but its increase as well. This intervention is a direct consequence of the development of energy harvesting technologies. The fact that a device has a variable energy supply changes the rules of the game completely. While before a node was supposed to last for a limited amount of time, now it can last more, virtually forever. Moreover more complex and heavier cryptographic solutions can now be employed, given that the device has sufficient energy to do this when the time comes. At a first sight it could seem that energy harvesting produces stronger

security and does not weaken it. Unfortunately that is not true as the author argues in [26], [28] and [27]. To give an example, Taddeo et al. in [60] present a EH-WSN system that provides QoS in secure and energy preserving communication. Here messages are defined by two properties:

- Priority: from low to high.
- Security: from low to high, with different combinations of properties (e.g. confidentiality, integrity etc.).

A message is sent only when the node has the right amount of energy to do so (with respect to all other messages). Highest priority messages are sent first, with the most secure policy possible according to the energy available. Now by exploiting *Energy Control* or *Energy Reduction* the attacker could force the device to send out $HighPriority/HighSecurity$ messages with a lower security characteristic, effectively changing the message from $HighPriority/\mathbf{High}Security \rightarrow HighPriority/\mathbf{Low}Security$, violating the security policy. This is a small but powerful example of what can happen if a system is not securely designed against a correct attacker model.

Interventions are modular. The attacker may have a subset of them or the whole, depending on her strength, and each and every one of them comes with a cost[4].

After reading about interventions, it should be clear to the reader that each and every one of them represents the result of a series of atomic actions that together lead to the corresponding intervention. These actions are different for every system, e.g. approximate localisation of a device through *Eavesdropping* in a BAN is very much different from that in a WSN: BAN are short range, highly localised systems, therefore when you have detected a device you are in its proximity, more notably the whole system is there; whilst when it comes to WSN, localising a specific node requires more complex operations (e.g. triangulating the signal) and, since signal spreads through a wider area, even more time depending on the resources at the attacker's disposal. Thinking about more peculiar interventions, *Energy Control* and *Energy Reduction* depends very much on the type of energy source the devices have.

2.2.3 Relationships Between and Within Dimensions

In the first part of this section the author has defined the attacker's dimensions, namely the space and time domain she moves over and the actions she can

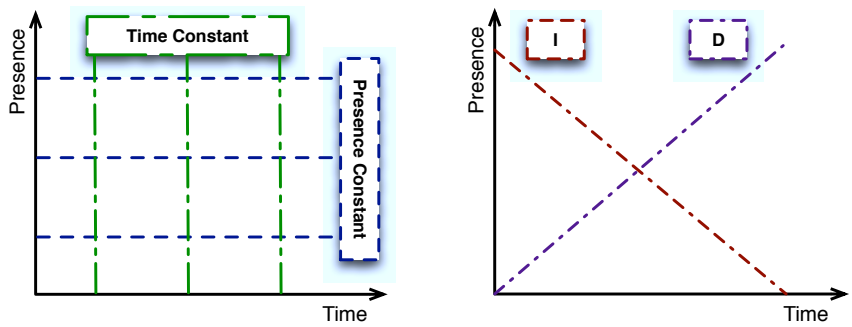
[4]See Section 2.2.4.

perform. What remains to be discussed is how dimensions interact with each other and what assumptions or implications each element may have with respect to the attacker's identity. Moreover interventions deserve a further analysis for several reasons:

- They constitute the most complex dimension.
- They describes the actual abilities of the attacker (i.e. what she can and cannot do).
- Some interventions may very well depend on others.

Presence and Time

Presence and Time can be analysed either independently (i.e. the value of one is fixed while the other changes), or by relating ones value to the others. This concept is shown in Figure 2.4.



(a) (b)

Figure 2.4: Presence and Time Relation

In the case of Figure 2.4a, Presence and Time are unrelated to each other. With respect to Presence, going from *local* to *distributed* and *global* makes the attacker more and more powerful since she is able to exert her influence over a larger part of the system. On the other hand, with respect to time, going from *short-span* to *long-span* enables the attacker to more complex attacks[5], and gives her a better knowledge about the system and the data it encompasses.

Looking at it from another angle (Figure 2.4b), Presence and Time can be related in an inversely (I) or directly (D) proportional fashion. Specifically:

[5]In this analysis *multiple-spans* are enabled by *long-spans*.

- Direct proportionality between Time and Presence is the most obvious relation: the wider the area the attacker influences, the more time she has at her disposal. Extending the area of influence requires a stronger attacker, therefore it seems only natural that she is allowed for more time too (Figure 2.4b line "D").
- On the other hand, considering that an attacker might want to go undetected, influencing a large part of a system (possibly the whole) for a long time, would most likely be noticed right away. Therefore the attacker might prefer another approach: either influence a big area for a short time or do it on a small area for a long time (Figure 2.4b line "I").

At this point it is possible to make a first remark about the attacker's strength. It is shown in Figure 2.5: the strength grows together with Time and Presence. That means that an attacker needs to be stronger if she wants to expand is temporal or spatial influence.

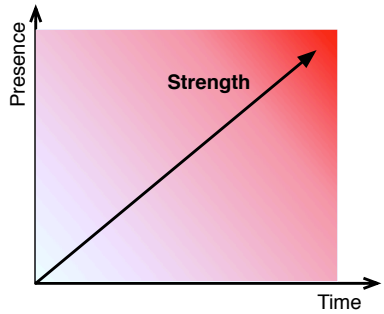


Figure 2.5: Attacker's Strength with Respect to Presence and Time

This strength can be quantified in terms of resources the attacker has at its disposal, but we will see more about this in Section 2.2.4.

Interventions

While reading about Interventions, the thought that Interventions are not independent between one another has probably crossed the mind of the reader. As a matter of fact some of them present similarities (e.g. *Energy Reduction* and *Energy Control*): one intervention may well be a mandatory step for the realisation of another (e.g. localisation through *Eavesdropping* and *Destruction*) or a consequence of another. Moreover the strength of an intervention could be augmented by another one. For example having *Energy Reduction* together

with *Eavesdropping* (which enables localisation), strengthens the first one by allowing the attacker for local energy reduction. In fact, without localising the device, the attacker would have less means to carry out *Energy Reduction* (e.g. generate false traffic).

There are three relations that link together interventions:

- Consequence: an intervention is the consequence of another one.
- Strengthen/Support: an intervention augment the strength of another one by supporting it.
- Imitation: an intervention have an effect that imitate another intervention without the need for this to be carried out.

Table 2.1 details the connection between each intervention. Each intervention of the first column is linked to one in the first row by one of the three defined relations.

	Destruction	**Eavesdropping**	**Data Knowledge**	**Disturb/Partial Data Modification**	**Full Data Modification**	**Reprogramming**	**Device Injection**	**Energy Reduction**	**Energy Control**
i. *Destruction*	/						S		
ii. *Eavesdropping*	S	/	C	S	S	S	S	S	S
iii. *Data Knowledge*			/			S	S	S	S
iv. *Disturb/Partial Data Modification*				/			S	S	S
v. *Full Data Modification*					/		S	S	S
vi. *Reprogramming*	I	C	S	S		/	S	S	S
vii. *Device Injection*		S	S	S			/	S	S
viii. *Energy Reduction*	I			I				/	
ix. *Energy Control*	I			I					/

S - Strengthen/Support, I - Imitation, C - Consequence

Table 2.1: Relationship between interventions

To explain Table 2.1:

1. *Destruction*: can support device injection by favouring the insertion of a new device into the system in order to provide for the disappearance of the destroyed device.

2. *Eavesdropping*: has a relationship with all the interventions since is needed as a first step to discover devices, and then to monitor them as a further step. Specifically it can support *Destruction* by locating the device, the same goes with *Full Data Modification* and *Disturb/Partial Data Modification*. By monitoring the network, the attacker could gain information about the functioning of the system that she can then use to reprogram a device or to inject one in the network. Finally eavesdropping can support also *Energy Control* and *Energy Reduction*, depending on the specific implementation of energy saving algorithms. Furthermore if the attacker can make sense out of the data (i.e. they are not encrypted or the attacker can decrypt it), this could bring to *Data Knowledge*.

3. *Data Knowledge*: can support *Reprogramming* and *Device Injection* in the same manner that eavesdropping does. Similarly it supports also *Energy Control* and *Energy Reduction* as the memory could store data about the power and energy usage policy of the node.

4. *Disturb/Partial Data Modification* and *Full Data Modification*: can support *Device Injection* by forcing the system to accept a particular node. With respect to energy, to modify the data of a device could mislead the device into using its power in an anomalous manner, thus resulting in *Energy Control* or *Energy Reduction*.

5. *Reprogramming*: as a reprogrammed device behaves at the attacker's will, this intervention can be related to all the others. Specifically it imitates *Destruction* since a reprogrammed device, although physically the same, can be seen as a complete new device, thus the old one ceasing to exists. A reprogrammed device can relay messages, enabling *Eavesdropping*. Furthermore it can support all other interventions, apart *Full Data Modification* which requires physical access to the device.

6. *Device Injection*: very similar to reprogramming, has the same relationship with all the interventions apart *Destruction*: as the injected device is physically a new device, it does not imitate destruction as *Reprogramming* does.

7. *Energy Reduction* and *Energy Control*: these two interventions imitate *Destruction*, since they can deplete the energy of a device, and *Disturb/Partial Data Modification*, since changing the energy level of a device interferes with its normal operations.

2.2.4 Considerations over the attacker strength

In this section the author wishes to make some considerations and remarks about the attacker strength, so that the reader is aware of this aspect, specifically how the attacker's strength influences the model, and how much the design of a system strongly depends on the assumptions made about the attacker strength.

When it comes to design and implement security in a system, one has to consider two things:

1. The attacker model, which we extensively talked about in the previous part of this chapter.
2. The attacker strength and how this interacts with the model.

It is common practice to quantify the attacker's strength by the money she puts into resources (see [57, Chapter 7]). This is indeed a very convenient way since one does not have to focus on computing power, memory, time or other resources alone, but one assumes that these resources can be acquired somehow with a common mean (e.g. money). This method becomes very effective when it is possible to quantify exactly the strength needed for a specific attack. This is the case of cryptography and cryptographic keys, where the amount of resources needed to find a specific key can be physically expressed in terms of memory and time. Table 2.2, taken from [57], shows a classification of attacker's resources in terms of money and type of technology the correspondent attacker uses.

Attacker	Budget	Hardware
"Hacker"	0	PC(s)
	< \$400	PC(s)/FPGA
	0	"Malware"
Small Organisation	\$10k	PS(s)/FPGA
Medium Organisation	\$300k	FPGA/ASIC
Large Organisation	\$10M	FPGA/ASIC
Intelligence Agency	\$300M	ASIC

Table 2.2: Classification of the Attacker in Terms of Resources - Source [57]

Before considering the strength of the attacker with respect to her actions (i.e. atomic actions and interventions), it is only logic to start from the dimensions over which these actions are carried out: *Presence* and *Time*. In Section 2.2.3 the writer already gave a classification of the attacker strength in terms of *Presence* and *Time* now looking at Table 2.2 it is possible to make further considerations with respect to the type of attacker.

Considering presence this is the case where the physical and cyber features of the attacker part away from each other. Weak attackers (i.e. Hacker) surely do not have the resources to be in multiple physical locations at the same time, whereas strong attackers do (i.e. large organisations and intelligence agencies). On the other hand a weaker attacker could still get to have a global cyber control over a system.(e.g. through malware or botnets).

With respect to strength, time has to be analysed from two perspectives:

- Time as a measure of the period during which the attacker controls the system.
- Time as a measure of the physical time needed for an attack.

In the first case the strength of the attacker is directly proportional to the time, in the second case it is quite the opposite. For example to find a key back in 1996, an hacker needed 222 *days* to break a 45 *bit* key while an intelligence agency would have done it in 73 *days* for a 75 *bit* key (source [57],[17]). On the other hand thinking about controlling a system, the stronger the attacker, the longer the time she is able to exert her control.

After having considered strength against presence and time, interventions and atomic actions come along. The outcome of every intervention depends very much on the attacker's strength: it is closely linked with how much the intervention can benefit from an attacker that is stronger with respect to time or presence or both. In the case of an attacker who wants to perform traffic analysis, eavesdropping a network with a global adversary for a short time would be less effective than a local or distributed adversary, which is able to eavesdrop for a considerable amount of time. Similarly having a global attacker that exert *Energy Reduction* for a short time is definitely more threatening than one that does it locally for long time.

Ultimately is the attacker strength with respect to atomic actions that matters[6]. These are very much quantifiable since each action requires specific resources, depending on the system, which can be classified with respect to strength levels. Medium channel actions for instance, as well as physical, can be quantified on the hardware the attacker uses e.g.: type of antennas, computing power, memory dimension etc. Cryptographic actions instead require perhaps a different type of resources, from computing facilities up to laboratories (e.g. for power analysis).

Knowing about attacker strength and how it affects the model is very important when someone wants to perform a quantitative security analysis of a system, as the work done in this thesis aims to do. This will be clearer in Chapters 3 and

[6]Remember that interventions are composed by atomic actions.

4.

2.3 Case Studies: Qualitative Analysis

In this Section two case studies are presented, an insulin pump system (defined in [43]), and a WSN for area monitoring (defined in [52]).

2.3.1 The Insulin Pump System

The Insulin Pump system is composed of five wirelessly interconnected devices: the insulin pump, a remote control, a glucose meter, a continuous glucose sensor and a PDA. By looking at interventions and their interactions, *Disturb/Partial Data Modification* is the major threat for this kind of systems: by forging false commands an attacker would be able to modify insulin doses. *Eavesdropping* can lead to *Disturb/Partial Data Modification* (see Table 2.1), and with respect to *Eavesdropping* it is possible to: (a) listen on data by finding the frequency and get the suitable hardware (easy according to [43]), (b) read and understand data since no encryption is used.

Going further to *Disturb/Partial Data Modification*, it is possible to forge and send false commands to the system since: (c) the PIN is sent in clear, (d) the parameters of the CRC, which is an integrity check algorithm, can be easily found.

By looking at atomic actions, the designer would have been aware of these threats by considering *listen* for (a) and (b), and *inject* for (c) and *exploit crypto implementation* for (d) although CRC is not a cryptographic algorithm per se, anyway finding its parameters is similar to finding secrets of hash functions or digital signatures. The issue about the counter, found in [43], is not a major threat as long as (a,b,c,d) are addressed, on the contrary it is a normal practice to accept counter values higher than the last received one in remote radio control systems since it is very likely that a counter increases just because the remote control is triggered while the receiver is out of range.

Up to now, our approach helped us to find major threats which were already known in [43], but the analysis goes further on. Assuming the designer has fixed (a,b,c,d) by proper and efficient measures (e.g. ensuring confidentiality by using lightweight cryptography standards like [36]), our attacker model prompt us to explore other possibilities. Looking at *Data Knowledge* or at cryptographic

atomic actions, like Differential Power Analysis (DPA [37]), it could be possible for the attacker to get hold of cryptographic keys or achieve *Reprogramming* thus bringing back (a,b,c,d) all together. What we want to stress here is that 100% security can not be accomplished. It is only a matter of what level of security the designer wants to achieve. In this case, if for the attacker is impossible, or better very unlikely, to get hold of a device, than the designer could simply ignore further threats. Besides it would be strongly recommended that every insulin pump system has an independent set of keys, so that if one system is compromised, all others are still safe.

2.3.2 FleGSens, a WSN Area Monitoring System

The FleGSens system [52] is a WSN for monitoring trespasses in an area under surveillance. It is composed of two main kind of devices: gateways and sensors. Gateways receive the transmissions from the sensors, organise data and relay it to the control centre. Sensors are scattered throughout the surveilled area, they are equipped with an infrared sensor to detect movements.

The system makes use of AES-128 block cipher which is used only to authenticate messages, this means that data are sent in clear (as for the insulin pump) but they cannot be forged without knowledge of the key. Each sensor stores a key preshared with the gateways which is used to authenticate each message sent by the sensor.

Within FleGSens there are two protocols in place that were designed to strengthen the system both in terms of efficiency and security:

- Trespass Detection Protocol.
- Node Failure Detection Protocol.

The first one organises neighbouring nodes in small clusters which gather collective information about trespasses and send it in one transmission only. This is done for several reasons: to prevent one event triggering traffic flooding and to avoid false positives due for example to animals or similar sporadic events. To do this a sensor recognise a movement as a trespass only if two movements are detected within a variable time usually set to 10 seconds. Therefore if a sensor registers two movements more than 10 seconds apart, it disregards them.

The second protocol is needed to detect node failures, so that failed nodes can be repaired or replaced. It is important to note that this protocol would mark as failed also nodes that have been destroyed or whose battery has been depleted or taken off.

There are also other protocols among which localisation protocols, that enable sensor localisation to know where a trespass has happened, and routing protocols which establish multiple routes between sensors and gateways to ensure that messages reach the destination.

Looking at the interventions the attacker might carry out several threats: she could make her way through the network by depleting the energy of the nodes; this can be achieved by replying old messages, thus making nodes waste energy in the processing. Thinking about a more elaborate threat, the attacker might reprogram a device, getting hold of the cryptographic material. With a reprogrammed device she could then tamper with the routing algorithms, triggers false alarms or overload the communication system. She could also get the location of the nodes through the location protocol and exploit this information in some way. As messages are only authenticated and not encrypted, some of these threats could be achieved simply by eavesdropping the packets from safe distance.

2.4 Summary

In this chapter the author defined the attacker model. First a methodology based on the inductive approach was discussed. Specifically the attacker model is developed around WSNs, EH-WSNs and BANs. After the model is introduced. The attacker is defined by three dimensions: *Interventions*, *Presence* and *Time*. The relationship between the dimensions is then discussed in connection with the attacker strength. At the end of the chapter two case studies are introduced.

Chapter 3

Quantitative Modelling

In this chapter an implementation of the attacker model is presented. The first part of the chapter introduces the modelling language, the PRISM Language, while the second specifies how the model is implemented and how properties can be analysed within it.

3.1 Choosing a Suitable Modelling Language

In Chapter 2 the attacker model which this work is based on was introduced, the next step is to choose a proper modelling technique, namely a suitable modelling language and a model checker.

To achieve this, it is necessary to:

1. Draw a list of requirements that the model has to fulfil.
2. Investigate what classes of models fall into the requirements.
3. Find a suitable modelling language that models those classes.

that

Looking at the model, concerning the requirements, it is clear that:

- The three dimensions have to be explicitly or implicitly modelled.
- A way to quantify the security of a system, or its parts, should be provided.
- The attacker strength has to be taken into account.
- As some Atomic Actions have a probability of success (e.g. eavesdropping a transmission) the model should be able to implement this probabilities.

From this list there are two terms that are directly or indirectly underlined: quantitative and probabilistic. Looking at the dimensions: Time is a quantity, Presence is a quantity, the success of an intervention can be expressed in terms of probabilities. The security of a system can be quantified through the probability of violating one or more parts of it and with respect to the attacker a system provides different levels of security according to her strength (cf. Table 2.2). Furthermore as ubiquitous systems devices can be defined in terms of the technical resources or energy constraints they have, a quantitative model would be able to make use of these information. Finally one has to take into account that different systems require different levels of security: in [33] for example, a detection grid for cane toads is defined. This is a system used to monitor the whereabouts of cane toads for scientific purposes; clearly such a system does not have the same security requirements as systems for health care, sensor networks for military applications or border monitoring (e.g. [59]), therefore a suitable modelling language should allow for a quantitative analysis that can express a measure for the security level of a system.

Looking at the requirements we can consider four classes of models which could be useful to describe the type of systems which this work focuses on, namely Discrete Time Markov Chains, Continuous Time Markov Chains, Markov Decision Processes and Probabilistic Timed Automata.

Discrete Time Markov Chain

A Markov Chain is a mathematical system that models the transitions from one state to another. In a Discrete Time Markov Chain every transition accounts for a time unit, and time is a discrete integer value. The main property of Markov chains is called memorylessness, it means that every state of the chain depends only on the previous one. Furthermore Markov Chains have other properties depending on the type of system described, these are:

- Reducibility: a chain is irreducible if is possible to get from any state to any state, otherwise is said to be reducible.
- Recurrency: a state j is said to be recurrent if the probability of going back to it (possibly after an infinite time) is equal to 1. It is `positively` recurrent if the time to return to j is finite. A state j is an `absorbing` state

when is not possible to leave it. A state j is said to be `periodic` if there exists an integer k such that it is possible to return to j in a number of steps which is a multiple of k.

- Ergodicity: a state is said to be ergodic if it is not periodic and it is positively recurrent.

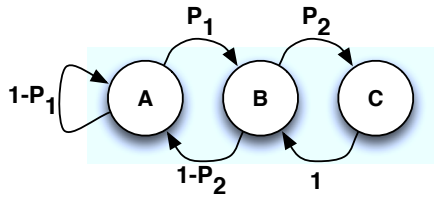


Figure 3.1: Discrete Time Markov Chain

Figure 3.1 shows an example of a DTMC. A probability of occurrence is assigned to each transition. Notably the probabilities of the transitions originating from the same node need to sum to one.

Continuous Time Markov Chain

A Continuous Time Markov Chain models stochastic processes where time is continuous and the time spent in a state follows an exponential distribution (see Formula 3.1).

$$f(t, \lambda) = \lambda e^{-\lambda t}, t \geq 0 \tag{3.1}$$

Transitions between states are specified by a rate λ which is a parameter for the exponential distribution; it expresses the number of transitions per time unit (e.g. seconds or minutes).

Figure 3.2 shows an example of a CTMC. Whenever there are two or more transitions originating from a state, the time spent in a state follows an exponential distribution where the *rate* is given by the sum of all outgoing transition rates: the time spend in state B for example is given by an exponential distribution with parameter $\lambda_2 + \lambda_4$. It is still possible to compute a probability of taking a

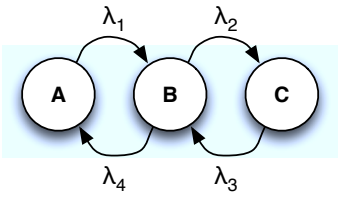


Figure 3.2: Continuous Time Markov Chain

transition, this is given by:

$$P[T_k] = \frac{\lambda_k}{\sum_{j=1}^{n} \lambda_j} \tag{3.2}$$

where T_k is transition k. Taking the CTMC in Figure 3.2 for example, the probability of going from B to A is given by[1]:

$$P[t + \epsilon = A | t = B] = \frac{\lambda_4}{\lambda_2 + \lambda_4}$$

Memorylessness still applies: given the state at a time t, the evolution of the system is independent of the events which happened before t.

Markov Decision Process

A Markov Decision Process is a formalism used to model systems that show both probabilistic and non-deterministic behaviour.

"Nondeterminism is an essential tool to capture several different aspects of system behaviour:

- *unknown environment*: if the system interacts with other components whose behaviour is unknown, this can be modelled with non-determinism;
- *concurrency*: in a distributed system comprising multiple components operating in parallel, non-determinism is used to represent the different possible interleavings of the executions of the components;

[1] In the equation $t + \epsilon$ with ϵ very small is used to represent a point in time after the transition from the state the system was at time t.

- *underspecifcation*: if certain parts of a system are either unknown or too complex to be modelled efficiently, these can be abstracted away using non-determinism."[2]

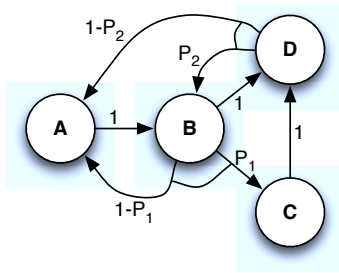


Figure 3.3: Markov Decision Process

An example of an MDP is shown in Figure 3.3. It is similar to a DTMC, as time is discrete (i.e. a transition happens after one unit of time is passed) and probabilities are assigned to transitions. The only difference is the non-determinism, which is expressed by multiple sets of choices coming out from a state, within each set the probabilities sum to one. In Figure 3.3 for example, originating from state B there is a non-deterministic choice between states (C,A) with probability $(P_1, 1 - P_1)$ or state D.

Modelchecking MDPs takes two steps: first all non-determinisms has to be resolved, then the model is checked as a DTMC. How non-determinism can be resolved is shown in Section 3.2.

the last state. over the last k states.

Probabilistic Timed Automata

A Probabilistic Timed Automaton is a Markov Decision Process augmented with clocks and constraints on clocks. Clocks are real values that measure the passing of real time. There can exist multiple clocks and they can be independently reset. A state or a transition can be constrained to a specific range of values of a clock.

Figure 3.4 shows an example of a PTA: x is a *clock*, each state is labelled with a constraint on the time, it is possible to be in state A only when $x < 2$, a

[2]Quoted from [32].

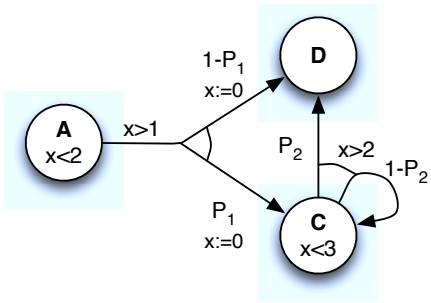


Figure 3.4: Probabilistic Timed Automata

transition from A to C and D can happen only when $x > 1$ and when the transition happens the clock is reset to zero ($x := 0$). Transitions are also labelled with probabilities.

To model DTMCs, CTMCs, MDPs and PTAs the writer chose the *PRISM* language and model checker. PRISM is known for its efficiency, the complexity of the models which it can address and its reliability; it has been widely used in the research community for diverse applications, ranging from real-time systems, communication and multimedia protocols, to game theory, power management, biology and security. It has a well structured property language which inherently supports time, both discrete and continuous (depending on the model class). It was for these reasons that the writer decided to use it in this work.

Compared to other modelling languages, in the literature there is another one worth mentioning in comparison with PRISM: MODEST [18]. MODEST is a modelling language for stochastic timed systems. "It combines concepts such as non-determinism, probabilistic branching, real time and continuous probability distributions in an orthogonal way such that precisely defined subsets of the language correspond to well-known models that are amenable to model-checking, for example Probabilistic Timed Automata (PTA) or Continuous-time Markov Chains (CTMCs)."[3]

The positive feature about modest is that properties of the model, such as continuous time, non-determinism etc. can be added orthogonally to the model, then depending on the combination of these, the model has to be checked with a suitable tool. This results in a complex modelling language to handle and also in the need of multiple tools. Furthermore MODEST makes use of PRISM and the PRISM language to check some models.

[3]Quoted from `http://www.modestchecker.net/`.

Hence PRISM was chosen for several reasons:

1. Models examined in this work can be modelled with PRISM: it seems natural to use PRISM directly so that models can be built and checked in a more efficient, complete and coherent way.
2. PRISM has its own tool which provides a comprehensive framework for building the model, its analysis and modelchecking.
3. PRISM provides for a well structured and defined use of rewards structures[4] which are crucial for the analysis of the model.

3.2 Introducing PRISM

PRISM [42] is a probabilistic model checker, a tool used for formal modelling and analysis of systems which exhibit a probabilistic, non-deterministic and real-time behaviour. It supports Markov models in the form of Markov decision processes (MDP), discrete-time Markov chains (DTMC), continuous-time Markov chains (CTMC) and probabilistic timed automata (PTA). The model is defined using the PRISM language, a state-based language based on the Reactive Modules formalism of Alur and Henzinger [5]. After the model has been specified, desired properties can be analysed. Such properties can express questions like: "What is the probability that the system is attacked before data are successfully transmitted?", or "What is the probability of the system being secure as long as the session is active?". Such properties express quantitative measures and therefore the model needs quantitative data as input, which is taken from real systems specifications, or by modelling some parts of the system (see Chapter 4).

When a PRISM model is built it is characterised by three parameters:

- **States**: the total number of states the model has.
- **Init states**: the number of initial states of the model.
- **Transitions**: the number of transitions between defined states.

These express the size of the model and the feasibility of its analysis. They can easily grow exponentially depending on the implementation; this fact has to be taken into account if one wants to define a model and check it against some properties: if a model is too big then its modelchecking is not feasible. In this case the model can be simplified by expressing it in another set of spaces and transitions, or by performing statistical model checking. This is achieved a sampling technique: a large number of random paths through the model are

[4] These will be introduced in Section 3.2.

generated and the result of the given properties is evaluated on each run giving an approximately correct result. For a detailed description of statistical model checking please refer to [49].

As PRISM models express probabilistic and non-deterministic behaviour, it is not possible to define exhaustively the model presented in Chapter 2, simply because parts of the model have non-stochastic, unpredictable states. This becomes clear when you think about reprogramming: PRISM can tell what is the likelihood of a device/node being reprogrammed, but after that the outcome of this action is obviously not predictable. On the other hand the purpose of this work is to express the probability of a device being reprogrammed; after that, other means of analysis need to be used to deal with this.

With respect to DTMCs, CTMCs, MDPs and PTAs, there are some remarks to make on how PRISM models them:

1. DTMCs and MDPs: time is a discrete integer value.
2. CTMCs and PTAs: time is a continuous real value.
3. DTMCs and MDPs: transitions are assigned with probabilities, PRISM checks whether these sum to one.
4. CTMCs: transitions are assigned with rates. The sum of the rates of the transitions originating from the same node define the time spent in the node itself, following the exponential distribution in Formula 3.1.
5. PTAs: transitions are assigned with probabilities and clock constraints which identify the region of time where the transition happens.

Finally about non-determinism, PRISM resolves it through the use of *Adversaries*. An adversary is a resolution of the non-determinism of a model. An adversary can be:

- *Deterministic* or *randomised*: the choice is either determined or made at random.
- *Memoryless adversary* or *Finite-memory adversary*: each non-deterministic choice is based either over the last state only or over the last k states.

When checking MDPs, PRISM performs the analysis against all possible adversaries and then, depending on the type of property checked, it reports the minimum and the maximum probability of the event described by the property.

A PRISM model is composed of one or more modules which can interact with each other. A module is defined by a number of local variables, that identify its state, and a set of commands which specify its behaviour. The global state of the whole model is determined by the local state of each module. Modules can interact with each other by synchronising specific transitions, or by allowing a

transition given the local state of another module e.g. module A variable x can increase by one if and only if variable y of module B is true.

```
1  //definition of model type
2  dtmc || ctmc || mdp || pta
3
4  //global variables
5  global y : [1..10] init 1; //integer
6  global b : bool init true; //boolean
7
8  //constants
9  const int radius = 12; //integer
10 const double pi = 3.141592; //double
11 const double area = pi * radius * radius; //double
       value computed by the given formula
12 const bool yes = true; //boolean value
13
14 //module definition
15 module dummy
16         x:[0..6] init 0; //local variables
17
18 //Commands
19 []       x!=6 -> 1:(x'=x+1);
20 [reset] x=6  -> 1:(x'=0);
21 endmodule
22
23 //Formula
24 formula double_x = x*x;
25
26 //Command structure:
27 [label] guard -> rate:(update)
28 [label] guard -> probability:(update)
```

Listing 3.1: An Example of a PRISM Model

Listing 3.1 shows a simple PRISM model, it comprises:

1. Line 2: model type, it has to be either `ctmc`, `dtmc`, `mdp` or `pta`.
2. Lines 5-6: global variables (these can be changed by any module).
3. Lines 9-12: constants values.
4. Lines 15-21: definition of a module, it contains local variables and a set of commands.
5. Line 24: definition of a formula it comprises a name and an expression. It can be used anywhere in the code as shorthand for the expression.

The behaviour of module "dummy" is very simple, its state is identified by an integer variable `x` which increases from 0 to 6 every step and at 6 resets to 0. A command is composed by a label, a guard that expresses a condition that has to be satisfied, and a set of updates which occur with a specific probability or a rate (if it is a CTMC).

This work mainly concerns with DTMCs, MDPs and CTMCs, as these are the suitable techniques for modelling the most common scenarios in the attacker model this thesis deals with.

3.2.1 Properties and Rewards Structures

Once a model has been defined within PRISM, it can be analysed using property specifications that can be augmented with reward structures. A reward structure is simply a variable which increases or decreases by a user defined value. The same reward can refer to multiple states or transitions, and have different values for each one of them. In the case of *Energy Control* or *Energy Reduction* for example, a reward structure can be used to get a probabilistic measure of how many times the attacker managed to modify the energy level of a device or what its average energy level is when exposed to such an attacker.

Listing 3.2 shows an example of a reward structure. Looking at the first two rewards, imagine a model whose states are identified by a variable x, the model is a DTMC: the reward "time" counts the overall time spent in the system by summing the time spent in each state (multiplying it by one), the reward "time_by_x" counts the time spent in each state multiplied by the number of the state, this could express for example a computing process where the processing time is directly dependent to a variable.

For the third and fourth reward imagine a model where an eavesdropper is modelled, reward 3 counts the number of eavesdropped transmissions by counting the number of times the transition `[success_eavesdropping]` is taken. Reward 4 counts the number of trials the eavesdropper does by counting the successful eavesdrops plus the failures. Note that the failures are counted by passing through a state, this is possible since time is discrete (i.e. the model is either a DTMC or MDP). Reward 4 is an example of how a reward structure can be composed of different events (being in a state or performing a transition).

Once a model has been defined along with rewards structures, it can be analysed through the definition of *properties*. There are three type of properties identified by three operators:

```
// the model is a discrete time markov chain
dtmc

//Reward 1
rewards "time"
    true : 1;
endrewards

//Reward 2
rewards "time_by_x"
    true : x;
endrewards

//Reward 3
rewards "eavsdropped"
    [success_eavesdropping] true:1;
endrewards}

//Reward 4
rewards "trials"
    [success_eavesdropping] true:1;
    [fail] true :1;
endrewards
```

Listing 3.2: An example of a PRISM Rewards Structure

- **The P operator**: expresses properties over the probability for an event/state, or a set of events/states, to occur.
- **The S operator**: expresses properties over steady state probabilities (i.e. probability of being in a state or set of states in the long run).
- **The R operator**: expresses properties based on rewards structures.

Each operator is then completed by a path expression which defines (a). "Space": a set of states or transitions, (b). "Time": a time reference in which the property has to be computed. In this work we are mainly focused on P and R operators since in the long run every system can be compromised.

Listing 3.3 shows an example of properties, based on the rewards structures in Listing 3.2. Property 1 computes the probability of having at least one eavesdropped transmission in the first 5 time units (e.g minutes or hours). Property 2 is a reachability reward property: it computes the reward accumulated along a path until a specific state is reached. In this case it computes the expected number of eavesdropped transmissions until a failed eavesdropping

```
1  //T is an integer representing time
2  const int T
3
4  //Property 1
5  P=? [F<T atk_eavesdropped_sta=true]
6
7  //Property 2
8  R{"eavesdropped"}=? [F fail=true]
9
10 //Property 3
11 R{"trials"}=? [C<T]
```

Listing 3.3: An example of a PRISM Properties List

occurs. Property 3 computes the cumulative number of trials before time T.

A complete resumé of the PRISM Language is given in Appendix A.

3.3 Model Implementation

This section describes how the model is translated into Markov Models and implemented in PRISM. Interventions are directly modelled into PRISM modules as each intervention can be broken down to a series of step by step actions, that can be defined by a probability, or a rate. Presence and Time are modelled following a different approach. They are used to compute the transition probabilities for the interventions modules. This is done for a very simple but relevant reason: if presence and time were to be modelled explicitly the space state would grow exponentially,resulting in a model impossible to check. On the other hand, when analysing a system, is only reasonable to do some assumptions about the attacker e.g. what are her strengths in terms of presence and time, or what kind of action she is able to perform in a specific environment. With these assumptions the model size is small enough so that it can be checked. Furthermore an analysis performed against some specific values of presence and time shortens the time needed to analyse the system considerably.

The section is organised as follows: the first part describes how interventions are modelled while the second part discusses about time and presence.

3.3.1 Modelling Interventions

As PRISM makes use of modules it seems only natural to model each intervention as one independent module or an independent set of modules, so that when they have to interact, this can be done by synchronising commands or guarding the state of another module. The entire state space of each intervention module (IM) can be divided into four sets:

1. INIT: a set of initialisation states that express the enabled abilities of the attacker,
2. START: a set of states that mark the beginning of the Intervention,
3. ACT: a set of states which describe the intermediate actions or states that compose the intervention,
4. RES: a set of final states that represent the successful or unsuccessful fulfilment of the intervention.

Each IM is defined as a Markov Decision Process, where the non-deterministic behaviour is expressed by a choice between parallel intermediate actions which separately and independently contribute to the same intervention. Figure 3.5 shows an example of a state machine for a generic intervention module. Here states are generally labelled as INIT, START, ACT and RES. An INIT state may develop into two or more branches. Each branch begins with a START state. Furthermore a non-deterministic choice is highlighted: here probabilities independently sums to 1 ($P_1 + P_2$ and $P_3 + P_4$). It could also be that there are not intermediate states, and the intervention is simply successful or unsuccessful. This is for example the case for *Eavesdropping*.

The remainder of this section describes PRISM state machines for every intervention form. Whenever required a state is divided into two states $A_{i,S}$ and $A_{i,F}$ which respectively express *Success* or *Failure* of an intermediate action.

Please note that each set of transitions from a state to another has a set of probabilities which sums to one.

Destruction

Figure 3.6 shows the state diagram of the module. The first possible transition goes either from INIT to $A_{0,1}$ or $A_{0,2}$ (i.e. a non-deterministic choice). P_1 is

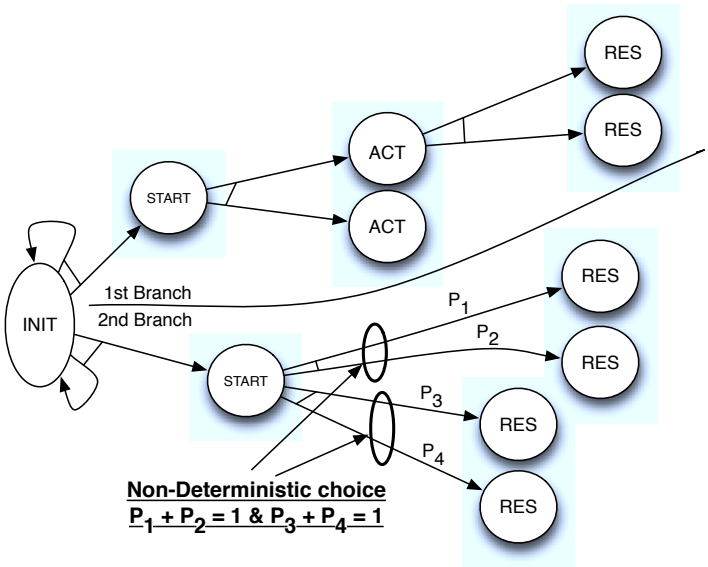


Figure 3.5: PRISM Intervention Module

the probability of localising a device, hence state $A_{0,1}$ expresses the physical localisation of a device. From there, $A_{1,S}$ expresses the successful physical destruction (e.g. crashing) of a device. Back to the INIT state on the other branch, P_2 is the transition probability that enables state $A_{0,2}$ which expresses the ability to destroy a device with other means which do not require exact localisation (e.g. electro magnetic pulses). This transition probability may very well be equal to 1 since either the attacker has this ability or she does not. Finally $A_{2,S}$ represent the successful destruction of a single device while $A_{3,S}$ the destruction of multiple devices. Clearly for each transition, $T_{0,1}$, $T_{0,2}$ and $T_{0,3}$, we must input a probability, which is strictly dependent on the system specifications, this is discussed in Chapter 4.

Eavesdropping

The state diagram of this intervention is shown in Figure 3.7. P_1 expresses the transition probability of being able to eavesdrop messages. As P_2 for Destruction, it could very well be 1, but it could also be that it is a real probability. This happens when the ability to eavesdrop comes from other actions ("source action" e.g. a reprogrammed device which forward all data to the attacker). In this

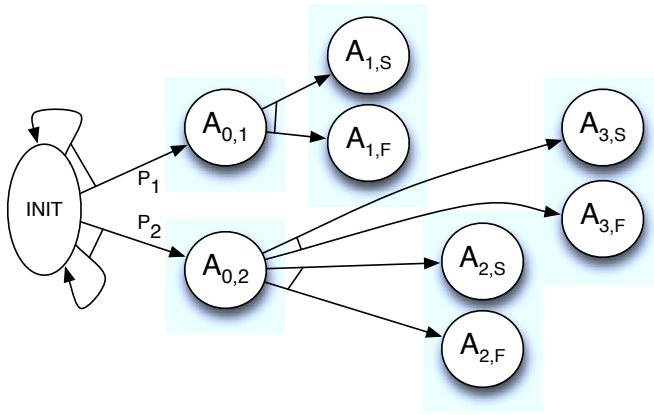


Figure 3.6: Destruction

case P_1 is the probability to successfully perform the "source action". State A_0 expresses the ability of eavesdropping messages while states $A_{1,S/F} \dots A_{n,S/F}$, represent the success/failure eavesdropping of messages in different modalities depending on the needs of the analysis. It could be for example that $A_{1,S/F}$ represent the eavesdropping of a single transmission, $A_{2,S/F}$ the eavesdropping of two consecutive transmissions etc. This kind of modelling is needed when the attacker needs to eavesdrop specific transmissions and not just a random one. Furthermore the overall probabilities of intercepting messages depend very much on the parameters of the system, e.g. whether the attacker/sources are mobile or not, and how they move in the space. In order to correctly compute such probabilities, devices and attacker movements need to be modelled as well. This issue will be thoroughly explained in Chapter 4.

Data Knowledge

As *Data Knowledge* expresses the fact that the attacker knows part or the whole content of the memory of a device, knowing where the device is (i.e. localising it) is a non-optional feature that the attacker needs to have. As shown in Figure 3.8, P_1 and A_0 are analogous to P_1 and $A_{0,1}$ for the Destruction IM in Figure 3.6.

A device usually relates to the surrounding environment to perform its actions, therefore memory is related to the environment as well. State $A_{1,S}$ expresses the successful knowledge of a value in the memory of a device, by learning it from

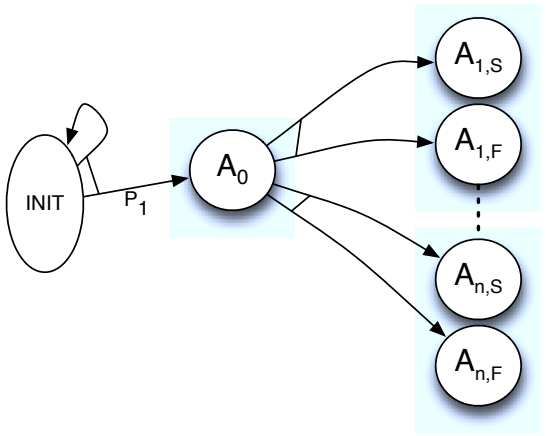


Figure 3.7: Eavesdropping

the surrounding environment, when this value depends on it (e.g. the device is an environmental sensor).

In the case where the attacker tries a more intrusive approach, $A_{2,S}$ represent successful access to the device, $A_{3,S}$ successful readout of the memory and $A_{4,S}$ the attacker knowledge of the value she is interested in. Note that there is a transition $A_{2,F} \rightarrow A_{2,S}$, which expresses the fact that the attacker may try to gain access to a device in an iterative fashion. If the device is tamper proof the transition probability of $A_{2,F} \rightarrow A_{2,S}$ is set to 0.

Disturb/Partial Data Modification

The state diagram is shown in Figure 3.9. P_1, which results in $A_{0,1}$, expresses again the probability of successfully localising the target device. $A_{1,S}$ expresses the fact that the attacker successfully changes the environment to feed a value into the device (e.g. in the case of a temperature sensor, the successful alteration of the local temperature) while $A_{2,S}$ expresses the fact that the value has actually been fed into the device memory. In fact altering the environment values does not imply directly that the value has been accepted into the device, hence the need for $A_{1,S}$ and $A_{2,S}$.

On the other branch P_2 and $A_{0,2}$ express the probability and capability of feeding the device with false data by injecting fake traffic into the network. $A_{3,S}$

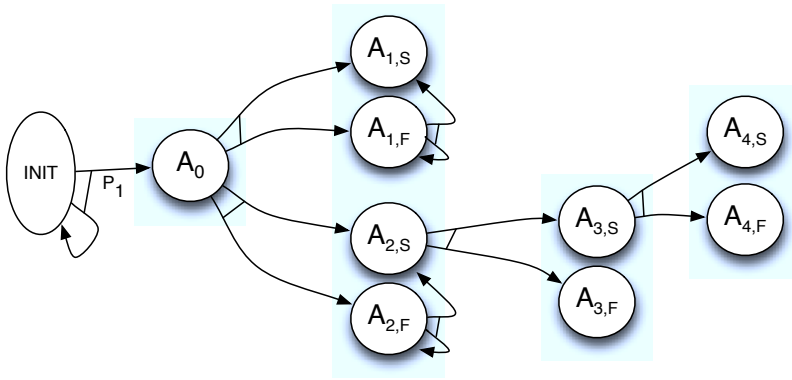


Figure 3.8: Data Knowledge

expresses the successful alteration of the target device memory. This branch of the state diagram has a broader and different scope than the previous one: values changed into the device memory can also relate to routing information or other parameters possibly related to security and communications throughout the whole network.

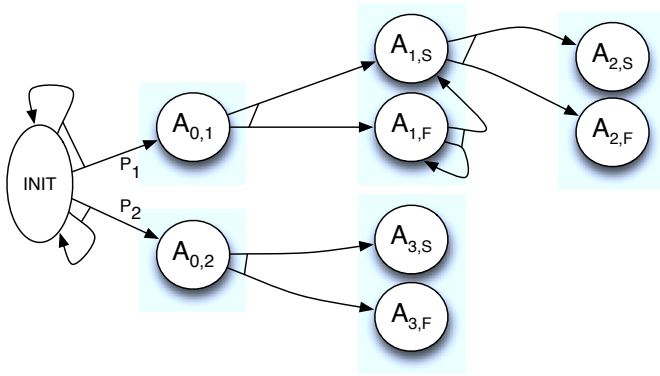


Figure 3.9: Disturb/Partial Data Modification

Full Data Modification

In contrast to the previous case, with *Full Data Modification* the attacker modifies the memory of a device by direct access to it, hence device-location is mandatory. Figure 3.10 shows the state diagram where P_1 and state A_0 as usual express localisation of the device. At $A_{1,S}$ the attacker gather access to the device memory, $A_{2,S}$ and $A_{3,S}$ represent respectively *read* access and *write only* access[5], while $A_{4,S}$ express *read/write* access. States $A_{5,S}$ and $A_{6,S}$ represent successful modification of the whole memory or a desired part of it, with the important difference that in $A_{6,S}$ is possible to verify that modification actually happened, since here the attacker has *read* access.

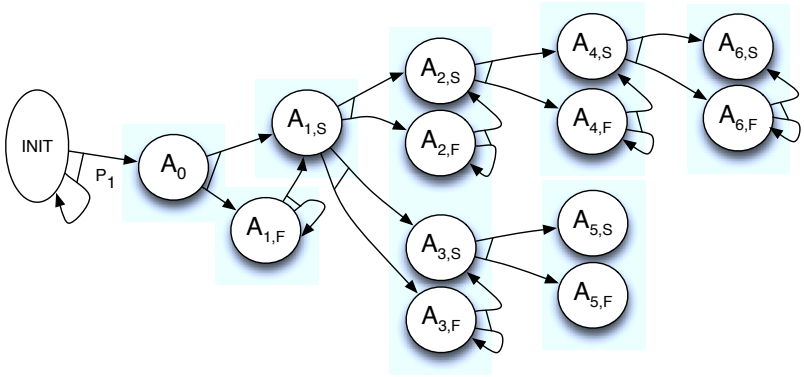


Figure 3.10: Full Data Modification

Transitions going from $A_{x,F}$ to $A_{x,S}$ states express the possibility for the attacker to try again. As in *Data Knowledge*, if the device is tamper proof or there are other countermeasures in place to block an action, some or all the probabilities of these transition could be set to 0.

Reprogramming

Is one of the most intrusive and dangerous interventions. P_1 and state $A_{0,1}$ (Figure 3.11) have the usual meaning of device location. $A_{1,S}$ represents successful access to the program memory of the device, and $A_{2,S}$ a successful reprogramming.

[5]There are cases when the attacker may have the possibility to write a value e.g. in a register, but not being able to read it.

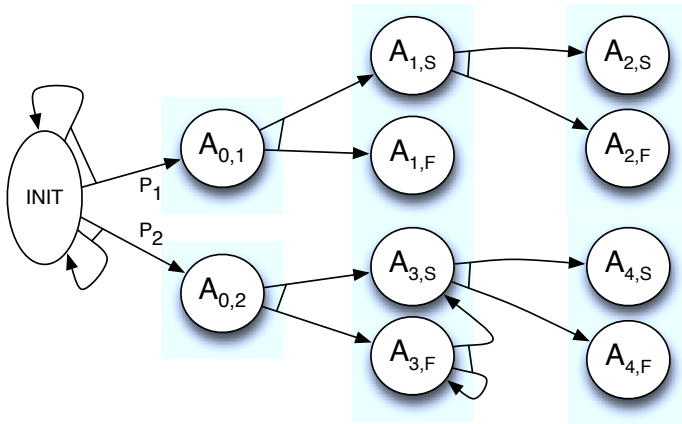


Figure 3.11: Reprogramming

P_2 and $A_{0,2}$ instead express the probability and the ability to reprogram a device remotely (e.g. through illegitimate actions like malware, or through legitimate ones like firmware upgrades). From there $A_{3,S}$ represent successful injection of a malware or remote access to the target device, and $A_{4,S}$ a successful reprogramming of it. Here transitions between $A_{x,F}$ and $A_{x,S}$ states have the same meaning as for the other interventions.

Device Injection

This intervention is rather simple to represent, since it depends strongly on the system, specifically on the policies for new devices and how are they accepted and correctly added to the system. The state diagram is shown in Figure 3.12, here P_1 and A_0 represent the probability and the ability of injecting new nodes and state $A_{1,S}$ a successful injection.

Energy Reduction

This is one of the two interventions that exploit the energy level of a device. *Energy Reduction* refers to the reduction of part or all the energy of a device. Figure 3.13 shows the state diagram for this intervention. P_1 and $A_{0,1}$ express the ability of tampering with a device while P_2 and $A_{0,2}$ express the opposite (i.e. no tamper capabilities). This is a non-deterministic choice as having tampering

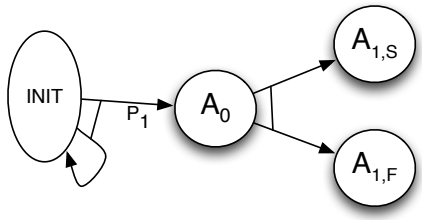


Figure 3.12: Device Injection

abilities is independent from being able to reduce energy without tampering, an attacker may very well possess both these abilities. $A_{1,S/F}$ represents success/-failure in locating the target device. From $A_{1,S}$ we go to $A_{3,S}$ which expresses the successful reduction of the energy of the target device by tampering with it (e.g. producing a power leak or simply taking out the battery), while, on the second branch, $A_{4,S}$ expresses the reduction of the energy by other means that do not involve tampering but need the device to be located, like forcing the device to transmit data but in a more surgical and local manner (e.g. continuously triggering of sensor measurements). If the device is not located (state $A_{1,F}$) then $A_{2,S}$ expresses the success in reducing the energy by other means (e.g. forcing the device to continuously transmit data by producing traffic elsewhere in the system).

Energy Control

For more complex systems which have a variable level of energy, *Energy Control* expresses the ability to control the energy level of the target. The energy of the device can also increase. It is the case for example of Energy Harvesting WSN.

The state diagram shown in Figure 3.14 begins like the one above, with $A_{0,1}$ and $A_{0,2}$ corresponding respectively to tamper/no-tamper capability, and $A_{1,S/F}$ success/failure in locating the target device. Then it goes on with $A_{2,S}$ which expresses success in controlling the energy by direct tampering, and $A_{3,S}$ where the attacker is able to exert this control with precision. $A_{4,S}$ has the same meaning as $A_{3,S}$ with the exception that this control is applied without tampering. This is for example the case of Energy Harvesting WSN where the attacker could stop or inhibit the energy harvesting by operating directly on the energy source.

The interventions that we have shown in this section are the basic modules for the whole attacker model. They can of course be composed according to the relationships they have (see Section 2.2.3).

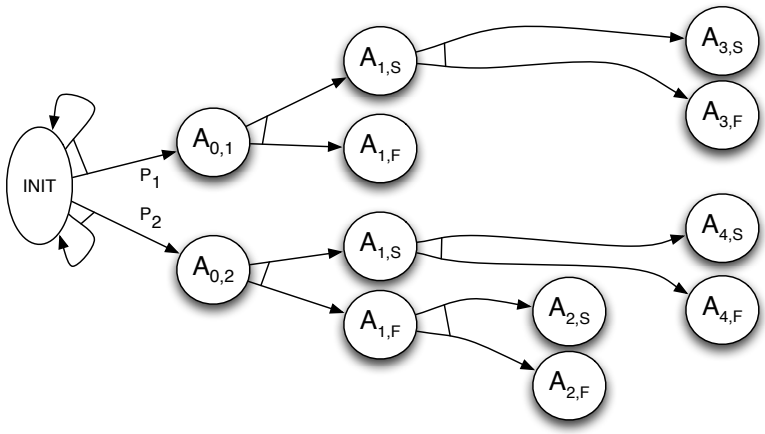


Figure 3.13: Energy Reduction

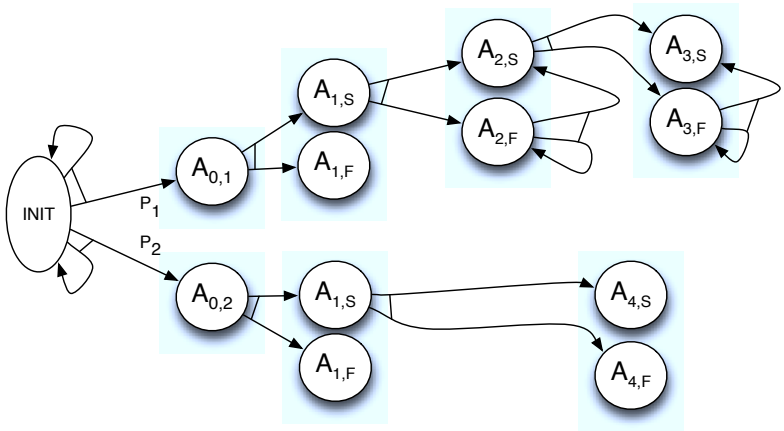


Figure 3.14: Energy Control

3.3.2 Time and Presence

Interventions are directly modelled into PRISM language by MDPs, but how is it possible to model time and presence in relation with interventions? As shown in Chapter 2, these two dimensions are strongly dependent on the attacker strength. Time and presence are needed to ultimately understand how long does the attacker take to successfully execute an attack and what is the extent of it.

We have already seen in Section 3.2 that PRISM inherently represents time as the real-time spent in a state if it is continuous, or when it is discrete as the number of transitions occurred from the initialisation of the system. This representation is useful when the *time* dimension can be directly related to the PRISM representation of time (Chapter 4 shows a case where this happens for the *Eavesdropping* intervention). When that is not the case, time is modelled through the use of a proper reward structure. It can be simple, i.e. express time through a unique coefficient, or more complex with several time coefficients depending on the attacker strength. Furthermore when defining the rewards structures, the transition probabilities have to be taken into account, as the time assigned to a transition depends on the probability and the strength of the attacker. This relationship can be written as a function:

$$time_{k,strength} = f(P[transition_k], strength) \tag{3.3}$$

where $P[transition_k]$ is the probability assigned to transition k and *strength* is the attacker strength. Assuming for example that the stronger the attacker is the less the time she takes perform an action, Figure 3.15 shows an example of time values for the same probability in correspondence with different level of the attacker strength.

Listing 3.4 shows the reward structure for the example above. Specifically the coefficients for `[transition_K]` refers to the values shown in Figure 3.15; whilst `[transition_N]` represents another transition different from K.

Differently from *Time*, *Presence* is modelled implicitly through the transition probabilities: depending on the assumptions about the attacker strength with respect to presence, different transition probabilities are assigned to the model. Therefore, to perform an analysis under different attacker presence values, one needs to change the relevant probabilities. For example in the case of eavesdropping, the probability of successfully eavesdropping packets changes with the presence of the attacker and of the underlined model used to compute eavesdropping probabilities. This will be explained thoroughly in Chapter 4.

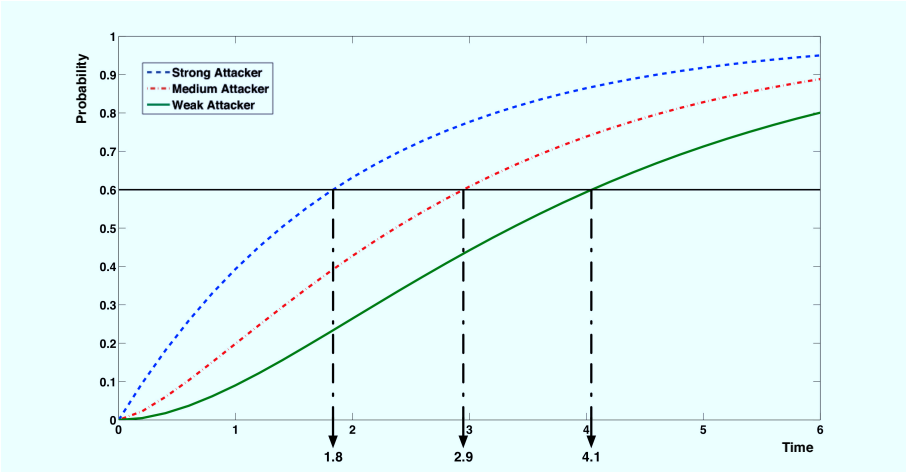


Figure 3.15: Cumulative Density Functions of a Transition Against Time with Different Attacker Strengths

Similarly to Formula 3.3, with respect to presence the transition probability can be expressed in terms of the attacker presence:

$$P[transition] = f(presence) \tag{3.4}$$

Therefore, considering Formulas 3.4 and 3.3 together, presence should be the first dimension that needs to be taken into account, then time and finally interventions.

3.4 Summary

This chapter proposes a quantitative implementation of the model defined in the previous chapter. The author first introduces the problem of choosing a suitable model language, Markov models are defined at this point. Then the PRISM language is presented, it is a state-based language augmented with reward structures. The modelling of the interventions, presence and time is then explained.

```
1
2  //Time Weak Attacker
3  rewards "TimeStrength_1"
4      [transition_K] true : 4.1;
5      ...
6      [transition_N] true : 6
7  endrewards
8
9  //Time Medium Attacker
10 rewards "TimeStrength_2"
11     [transition_K] true : 2.9;
12     ...
13     [transition_N] true : 5.5;
14 endrewards
15
16 //Time Strong Attacker
17 rewards "TimeStrength_3"
18     [transition_K] true : 1.8;
19     ...
20     [transition_N] true : 3;
21 endrewards}
```

Listing 3.4: Time Rewards for Different Attacker Strengths

Chapter 4

Probability Extraction

This chapter discusses the problem of computing transition probabilities for the quantitative model described in Chapter 3: this problem is called *"Probability Extraction"*. In the first part of the chapter the problem is introduced and described. The second part presents a way to extract probabilities for eavesdropping. Finally probability extraction for all other interventions is discussed.

Probability extraction is a crucial step in the analysis of a system, since the model defined in Chapter 3 is a quantitative model. After reading this chapter the reader is expected to understand what is the process behind the assignment of a probability or a reward time coefficient, within the PRISM model defined earlier, and what are the challenges the analyst has to engage in order to carry out the probability extraction.

4.1 The Problem of "Probability Extraction"

As the model presented in Chapter 3 is a quantitative model, in order to use it for the analysis of a particular system, the parameters specific to the model class (either transition probabilities or rates) have to be specified. Furthermore, whenever the analysis requires it, rewards coefficients have to be defined as well (see Section 3.3.2).

It is important to note that an analysis can be performed even though not all probabilities or rates are known. This can be done by setting unknown probabilities to one, in this way the final result will not be affected by the unknown probability; this is because the overall probability of a state is computed by multiplying one by one the probabilities of each transition belonging to the path that brings from the init state to the state itself. As for rates, it is slightly more complex since they define the overall time spent in the system, therefore there are two possible solutions:

1. Set the rate to an arbitrary high value so that the time spent in the state is relatively short.
2. Set the rate to an average value and take into account that the overall time in the system as a confidence interval dependent on the value input.

The first solution cannot be applied if there are multiple transitions coming out from the state since the transition with an higher rate is more likely to be chosen (cf. Formula 3.2), this would invalidate the model analysis for specific paths. When this happens, the analyst has to solve the problem case by case.

Extracting a probability is different between interventions and possibly even between transitions belonging to the same intervention. Therefore there is not a method that is general to do it for all interventions, quite the opposite. In the remainder of this chapter a method for extracting probabilities for *Eavesdropping* is explained; afterwards follows a discussion about probability extraction for other interventions. *Eavesdropping* was chosen, as most of the interventions needs localisation, which is based on eavesdropping. Furthermore eavesdropping is the most fundamental action for an attacker that wants to target an ubiquitous system, therefore its modelling is of primary importance.

4.2 Extracting Probabilities for Eavesdropping

In Ubiquitous Computing, mobility and ubiquity belong to each element of the system, therefore every model has to take into account these aspects and treat them accordingly. Eavesdropping is perhaps one of the most critical interventions with respect to mobility since here the spatial relation between attacker (i.e. listener) and source (i.e. device) has a huge influence determining whether or not eavesdropping is possible and to which extent. It is like when you want to eavesdrop a conversation between two people: if they are holding a position in a calm and quiet room, one can easily follow the conversation; on the other hand if they are walking throughout an open environment, filled with other sounds, even distinguishing words becomes difficult if not impossible. The literature is

full of mobility models which could be used for probability extraction, unfortunately all these models focus on asymptotic properties such as the expected length of a movement or the average speed of a device. For a survey on mobility models the reader can refer to [6].

These models are very useful when designing communication systems or drafting technology standards that have very stringent requirements in terms of Quality of Service (QoS) or throughput; on the other hand when it comes to compute values which concerns with the expected time to eavesdrop the first packet or the probability of eavesdropping two consecutive packets (this is of most importance e.g. for some localisation algorithms that rely on Round Trip Time), it is essential to know how the system behaves during a specific time window by modelling the evolution of a system step by step through time.

For this purpose a parametric model was developed in MATLAB [45]. MATLAB is a framework for numerical computation and visualisation which makes use of extended libraries, among which are libraries for mathematics and statistics. It has a programming language which resembles C/C++ in its structure although it is substantially different: it is interpreted by the MATLAB framework which deals with memory management, type assignments and similar so that the programmer can focus only on the algorithm, leaving typical problems, such as segmentation faults or exception handling, to the framework.

For the purposes of this work, MATLAB is used as a model generator for PRISM: all the parameters are inputted into MATLAB which at the end generates a PRISM model that can be then analysed with the PRISM tool (for details about the model generator please refer to Appendix B). This is very convenient as a PRISM model can easily reach 1500 lines of code for the mobility model alone. The model is a DTMC. Station mobility and transmission are modelled as independent concurrent modules; this means that station movements and transmission are independent. The model is defined by the following parameters:

- A 2D space whose dimensions are defined by x, y.
- n mobile/static sources.
- k mobile/static attackers.
- $step_x$ and $step_y$ defining the maximum distance a device can cover in one step.
- A vector which defines the eavesdropping probabilities at different distances.
- A data structure which defines how the device moves for every point in the space, according to a chosen probability distribution.
- $P(t_n = Tx|t_{n-1} = NoTx)$ the probability of transmission given that the source was not transmitting the moment before, it can be different for each source (from now on referred to as $P(Tx|NoTx)$).

- $P(t_n = Tx | t_{n-1} = Tx)$ the probability of retransmission, it can be different for each source (from now on referred to as $P(Tx|Tx)$).

The mobility can be either defined by $[start, end]$ points, or the movement can keep ongoing without solution; furthermore movements can either follow a specific path or be defined by a probabilistic distribution, specifically in the latter case the probabilistic distribution can be either independent on the position or biased towards a specific point in the whole space.

After the model has been built, we can check properties like: "What is the probability that attacker k eavesdropped on source n between time t_1 and t_2?", "In a $[start, end]$ walk what is the overall number/percentage of eavesdropped messages?". Questions like this are very important as we will see for the Insulin Pump case study.

The model is very versatile with respect to space, time and speed of the mobile stations. In choosing the value of these three variables the analyst has to apply to the following:

- The distance unit is not fixed, therefore depending on the model it can be interpreted as meters, tens of meters hundreds of meters and so on.
- The same goes with the time unit: it can be interpreted as seconds, minutes, hours etc.
- Space, time and speed of the devices are dependent on one another: for given two, the third is decided as $speed = distance/time$. Hence if you set for example the distance unit to $6m$ and you assume that the station moves at least at $2m/s$, then the time unit assumes the value if $3s$, which is the time needed by the station to cover $6m$.
- Mobile stations can have several speed values which are multiple of the $step_x$, $step_y$ value.

Furthermore the analyst should choose the distance unit according to the eavesdropping probabilities. In fact, to keep the number of transitions low, it is recommended for the distance probability vector to have a short length, hence if the maximum eavesdropping distance is $10m$ for example a good value for the distance unit should be $5m$ so that the eavesdropping probability vector contains only three values: those for eavesdropping probability at 0m, 5m and 10m. It is not always possible to do that as there are technologies whose transmissions are long range, nevertheless for short range transmissions this could be possible most of the time.

But how big a space and how many stations or eavesdroppers can be modelled? Table 4.1 shows the size of the mobility model in terms of states and transitions (cf. Section 3.2) for different parameters. The movement modelled in this case

is a random walk.

The columns are divided into four sets which differ with respect to one variable only:

1. Columns I to III differ for the number of static devices,
2. Columns I and IV to VII differ for the space dimension,
3. Columns VII to IX differ for the step size,
4. Columns X to XII differ for the number of mobile stations.

	I	II	III	IV	V	VI	VII
`Static/Mobile`	1/1	2/1	3/1	1/1	1/1	1/1	1/1
`Step Size`	1	1	1	1	1	1	1
`Space dim x;y`	10;10	10;10	10;10	20;10	20;20	30;10	30;30
`#States`	298	372	484	498	898	698	1 898
`#Transitions`	3 684	4 056	4 712	7 040	13 974	10 396	31 476

	VIII	IX	X	XI	XII
`Static/Mobile`	1/1	1/1	1/1	1/2	1/3
`Step Size`	2	3	1	1	1
`Space dim x;y`	30;30	30;30	5;5	5;5	5;5
`#States`	1 962	2 020	74	5 476	405 224
`#Transitions`	83 942	157 710	780	575 088	432 475 488

Table 4.1: Size of Mobility Model for a Random Walk

From this it is possible to draw some considerations:

1. The number of states and transitions increases linearly with the number of static devices by a small factor.
2. The number of states and transitions increases linearly with the product of x by y (i.e. the number of points in the space).
3. The number transitions increases linearly with the step size by a significant factor while the number of states grows by a very small factor.
4. The number of states and transitions grows exponentially with the number of mobile station.

The fourth consideration is very serious since it means that increasing the number of mobile stations makes the model impossible to check. On the other hand here the model considered is a random walk, where the mobility of a station is not limited in any way. As we will see for the case of the insulin pump, when the mobility of a station is specified along a specific path or it is constrained in a limited area, the model size is still small enough to be checked.

4.2.1 Eavesdropping the Insulin Pump: The Model

In Chapter 2, Section 2.3, we have introduced and described the Insulin Pump System according to the specifications given in [43]. The authors of the paper perform an attack on the system by capturing a single packet from the system. In order to investigate how real this threat is (i.e. what is the average time needed by an attacker to eavesdrop a packet) a mobility model which applies to a realistic scenario has been developed. The scenario taken into consideration is the following one:

> "A patient with an Insulin Pump System is at the hospital. An attacker, who wants to eavesdrop a packet from the patient's system, performs a survey moving either outside along the hospital perimeter or inside the hospital within the corridors."

The hospital modelled is the "RigsHospitalet" in Copenhagen, Denmark, one of the biggest hospitals in Denmark. Figure 4.1 shows a satellite picture of the hospital, along with its 2D geometry and dimensions.

The parameters of the model are set as follows:

- $x = 1 : 53, y = 1 : 25$, the unit space is $5m$.
- One mobile source, the patient.
- One mobile eavesdropper, the attacker.
- $step_x$ and $step_y$ are both set to 1, there is no reason to give the patient or the attacker the possibility of covering $10m$ with a step.
- The eavesdropping probabilities are set to $[0.95, 0.85, 0.1]$[1].
- A data structure which defines the mobility of a device for each point in the space, according to a chosen probability distribution.
- Regarding $P(Tx|NoTx)$ and $P(Tx|Tx)$ these are kept as variables, since the scope of the analysis is to assess how different values influence the overall result.

About eavesdropping probabilities, radio transmissions degrade as the distance between transmitter and receiver increases. A digital transmission is composed

[1] See page 65

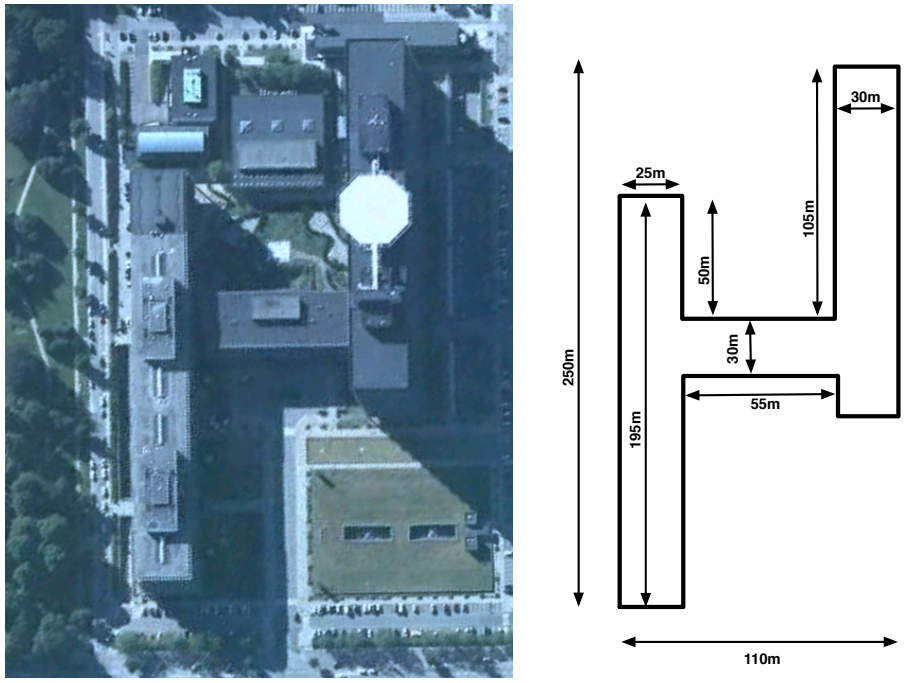


Figure 4.1: RigsHospitalet Geometry

of bits; a digital transmitter is able to decode a bit as long as the transmission is not too much degraded, thus a digital transmission is either received in its whole or it is not. Figure 4.2 shows the rate of successful decoding of a received packet against the power of the received signal, data are taken from [50]. As the quality of digital transmissions has a step shaped behaviour, three ranges were defined for the eavesdropping probabilities:

1. The attacker is in the very vicinities of the patient.
2. The attacker is within the transmission range of the patient.
3. The attacker is at the far edge of the transmission range of the patient.

In [43] it is said that a transmission from the insulin pump can be decoded up to 7 meters, therefore it is highly unlikely for an attacker to eavesdrop a packet from a bigger distance. Hence the eavesdropping probabilities were set to 0.95 for distances $d < 2.5m$ (i.e. half of the distance unit), 0.85 for distances

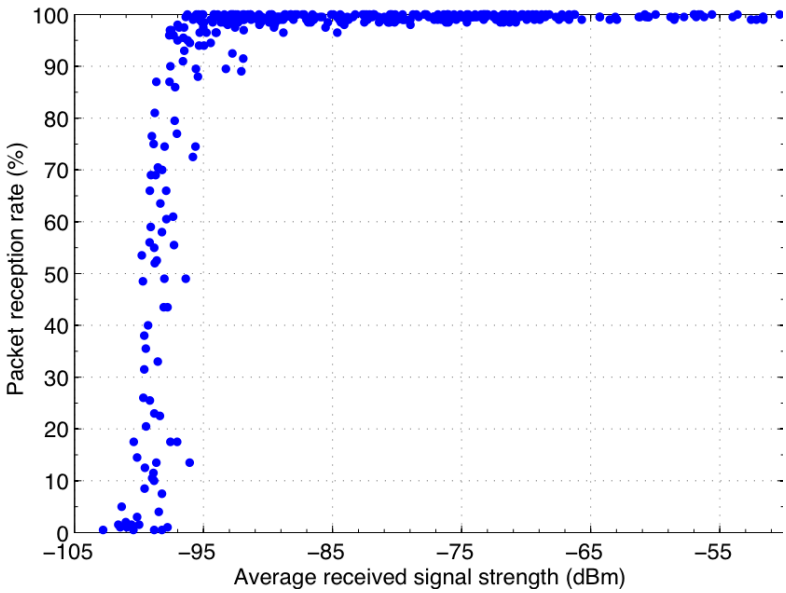


Figure 4.2: Success Decoding Against Signal Strength

$2.5m < d < 7.5m$ and 0.1 for $7.5m < d < 12.5m$, as a remark this is also the reason why the distance unit was chosen to be $5m$.

The simulations performed consider the following scenarios for the attacker and patient mobility:

- The patient:
  - is in a room of the hospital,
  - is waiting at the main entrance.
- The attacker:
  - is outside along the hospital perimeter,
  - is inside the hospital in public corridors.

Attacker and patient move according to different behaviours:

- The attacker pattern is shown in Figure 4.4; she performs two kind of movements: 1) outside performing a survey around the hospital 2) inside following a path within the corridors.

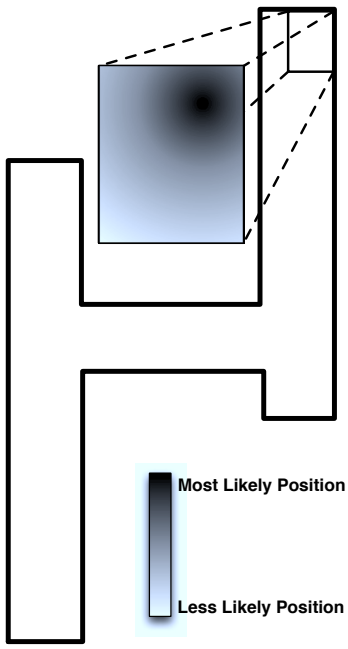


Figure 4.3: Patient Mobility Probability Density Function

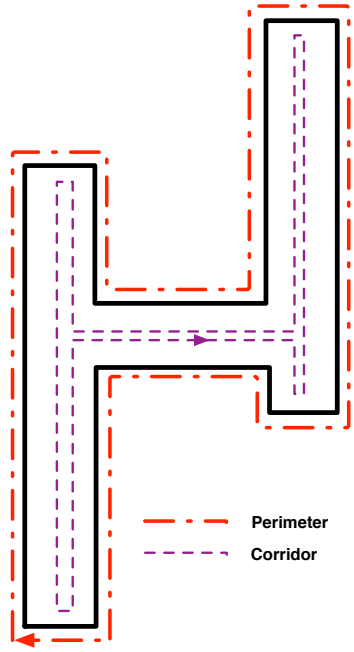


Figure 4.4: Attacker Mobility Pattern

- Differently the patient moves accordingly to a 2D normal distribution biased towards a focal point, this is the point where the patient is more likely to be. Figure 4.3 shows the patient mobility in a part of the hospital.

The PRISM model is shown in Listing 4.1. The listing samples the main parts of the model, specifically the PRISM modules, constants, formulas and reward structures. The listing contains also a short description of what each module does in the form of comments, these are introduced by //.

```
1 dtmc
2
3 const double p_tx_notx;
4 const double p_notx_notx = 1-p_tx_notx;
5 const double p_tx_tx;
6 const double p_notx_tx = 1-p_tx_tx;
7 const double p_eaves_0 = 0.95;
8 const double p_eaves_1 = 0.85;
```

```
9  const double p_eaves_2 = 0.1;
10
11 // Module that describes the movement of the Patient
12 module src1_movement
13     x_src1:[1..53] init ...;
14     y_src1:[1..25] init ...;
15     ...
16     [tick]  x_src1 = 4 & y_src1 = 21 ->
           0.023:(x_src1'=3) & (y_src1'=20) +
           0.036:(x_src1'=3) & (y_src1'=21) +...;
17     ...
18 endmodule
19
20 // Module that models the communication of the
       Insulin Pump
21 module src1_comm
22     tx_src1:bool init false;
23     [tick]  tx_src1 = false -> p_tx_notx:(tx_src1' =
           true) + p_notx_notx:(tx_src1' = false);
24     [tick]  tx_src1 = true -> p_notx_tx:(tx_src1' =
           false) + p_tx_tx:(tx_src1' = true);
25 endmodule
26
27 // Module that describes the movement of the Attacker
28 module eave1_movement
29     x_eave1:[1..53] init ...;
30     y_eave1:[1..25] init ...;
31     ...
32     [tick]  x_eave1=10 & y_eave1=21 ->
           1:(x_eave1'=11) & (y_eave1'=21);
33     ...
34 endmodule
35
36 // Module that model the eavesdropping of a
       transmission
37 module eave1_intercept_1
38     src1_coughtby_1:bool init false;
39     [tick]  tx_src1 = true & src1_coughtby_1=false &
           src1in_range_of1=2 ->
           p_eaves_2:(src1_coughtby_1' = true) +
           1-p_eaves_2:(src1_coughtby_1' = false);
40     [tick]  tx_src1 = true & src1_coughtby_1=false &
           src1in_range_of1=1 ->
           p_eaves_1:(src1_coughtby_1' = true) +
```

```
        1-p_eaves_1:(src1_coughtby_1' = false);
41      [tick]  tx_src1 = true & src1_coughtby_1=false &
        src1in_range_of1=0 ->
        p_eaves_0:(src1_coughtby_1' = true) +
        1-p_eaves_0:(src1_coughtby_1' = false);
42      [tick]  src1_coughtby_1 = false &
        src1in_range_of1>2 -> 1:true;
43      [tick]  tx_src1 = false & src1_coughtby_1 = false
        & src1in_range_of1<3 -> 1:true;
44      [cought11]  src1_coughtby_1=true ->
        1:(src1_coughtby_1' = false);
45  endmodule
46
47  //Formula that computes the distance Between attacker
    and Patient
48  formula src1in_range_of1 = max((x_eave1-x_src1>0
    ?x_eave1-x_src1:x_src1-x_eave1) ,
    (y_eave1-y_src1>0 ?y_eave1-y_src1:y_src1-y_eave1));
49
50  rewards "time"
51      [tick] true : 1 ;
52  endrewards
53
54  rewards "e1_eavesdropped_s1"
55      [cought11] src1_coughtby_1=true : 1 ;
56  endrewards
```

Listing 4.1: The Insulin Pump Eavesdropping PRISM Model

There are some remarks that should be made about the model listing:

- Constants are left undeclared, in this way the model can be checked against several values of the same constant.
- Most commands are labelled `[tick]`: this synchronises the commands so that transmitting and eavesdropping happen together with movements.
- Time is counted by a proper reward which counts the `[tick]` transitions.
- `rewards ''e1_eavesdropped_s1''` counts the eavesdropped packets.
- Formula at Line 48 computes the distance between the eavesdropper (i.e. the attacker) and the source (i.e. the insulin pump).

The sizes of the model are shown in Table 4.2.

	Perimeter	Corridor
`Space dim x;y`	53;25	53;25
`#States`	32 220	29 184
`#Transitions`	476 968	426 432

Table 4.2: Size of Mobility Model for the Insulin Pump Case

4.2.2 Eavesdropping the Insulin Pump: Model Checking

Given the model, any kind of properties can be checked against it, these can be properties regarding the number of consecutive eavesdropped packets or on the expected time to intercept one. As the security threat underlined by [43] needs just one eavesdropped packet for the attacker to be able to decode the PIN of the device hence enabling him to hijack it, the model was checked against the property shown in Listing 4.2, which computed the expected time to eavesdrop the first packet.

```
R{"time"}=? [ F src1_coughtby_1=true ]
```

Listing 4.2: Property: Time to Eavesdrop First Packet

Before showing the results and commenting on them, it is only natural to predict what the result should be. The probability of eavesdropping is closely linked with the probability that the source is transmitting. Figure 4.5 shows the DTMC which models the transmission of the Insulin Pump. The probability of transmitting (i.e. being in state "Tx") depends both on the values of $P(Tx|NoTx)$ and $P(NoTx|Tx)$, which is equal to $1 - P(Tx|Tx)$.

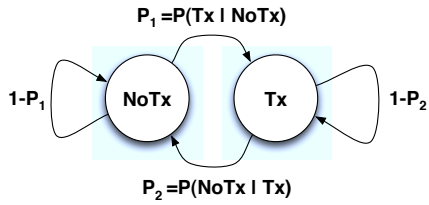


Figure 4.5: Transmission Module DTMC

The probability of being in state "Tx", π_{Tx}, can be computed by solving the equation system:

$$\begin{cases} \pi_{Tx} P(NoTx|Tx) = \pi_{NoTx} P(Tx|NoTx) \\ \pi_{Tx} + \pi_{NoTx} = 1 \end{cases}$$

which results in:

$$\pi_{Tx} = \frac{P(Tx|NoTx)}{P(Tx|NoTx) + P(NoTx|Tx)} \tag{4.1}$$

Figure 4.6 shows the values of π_{Tx} as a function of $P(Tx|NoTx)$ and $P(Tx|Tx)$. The overall probability of transmission grows with $P(Tx|NoTx)$ as[2] $1 - e^{-kx}$ with $1 \leq k \leq 4$, and exponentially with $P(Tx|Tx)$ as[2] e^{x-k} with $1 \leq k \leq 4$.

Figures 4.7 and 4.8 shows the evaluation of the property shown in Listing 4.2 for values of $P(Tx|NoTx)$ and $P(Tx|Tx)$ going from 0 to 1. Specifically Figures 4.7 shows the case where the attacker is performing a survey along the perimeter while Figures 4.8 the one where she is walking through the hospital corridors. Figures labelled as (a) refer to the case where the patient is near the window while the ones labelled as (b) to the case where he is in the middle of the room.

From the figures it is possible to see that when the patient is near the window, the expected time to get the first packet is less than when the patient is far from the window, this in the perimeter case. The corridor case present is the opposite behaviour the expected time is greater when the patient is near the window. This of course has to be expected, nevertheless there is a very noticeable result worth mentioning: the time to first eavesdrop strongly depends on $P(Tx|NoTx)$, that is when the device transmits the first packet of a transmission. This is somehow counter intuitive if you look at the transmission probability of Figure 4.6. Furthermore if $P(Tx|NoTx) \gtrsim 0.4$, the value of $P(Tx|Tx)$ more or less has no influence on the result. Hence, as an immediate result, if the constructor wishes to inhibit the hijack of the insulin pump he should choose some appropriate values for the transmission probabilities, focusing on the $P(Tx|NoTx)$ parameter.

In order to study how the probabilities of eavesdropping a packet within a given time grows with respect to $P(Tx|NoTx)$ and $P(Tx|Tx)$ separately, the model was checked against the following property:

[2]The equation is an approximate fit.

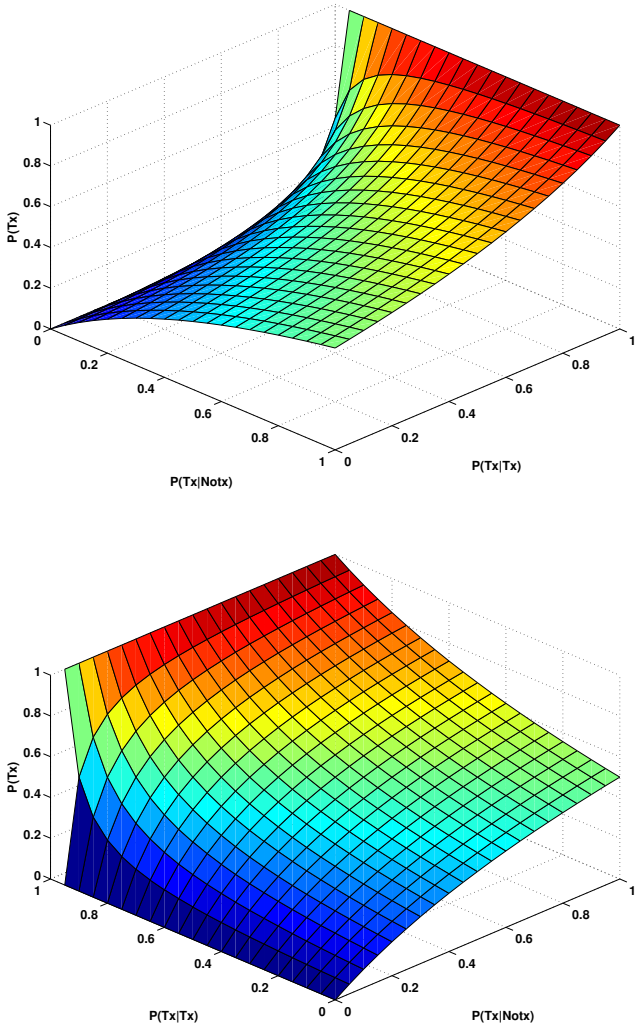
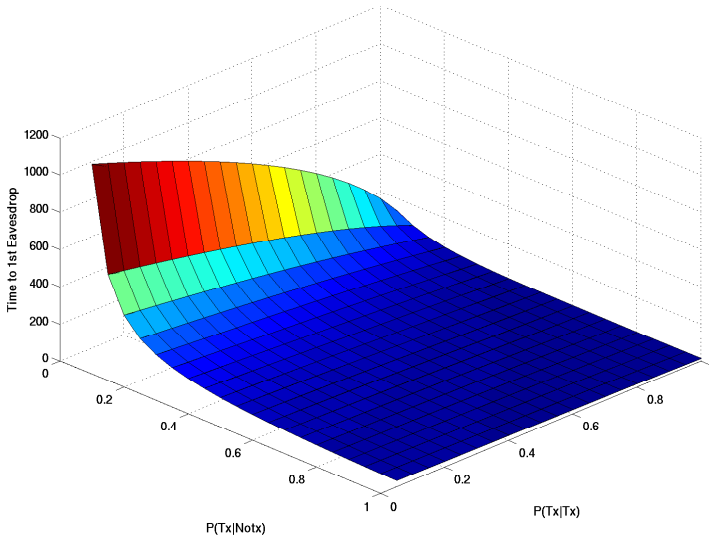
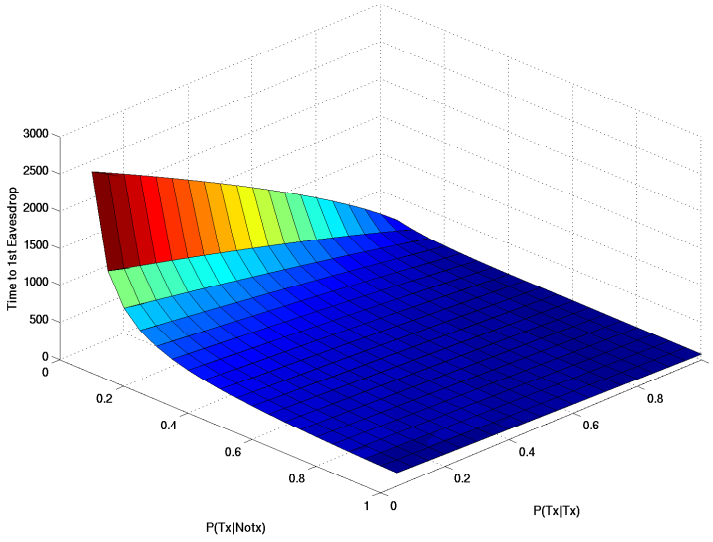


Figure 4.6: Transmission Probability of Insulin Pump

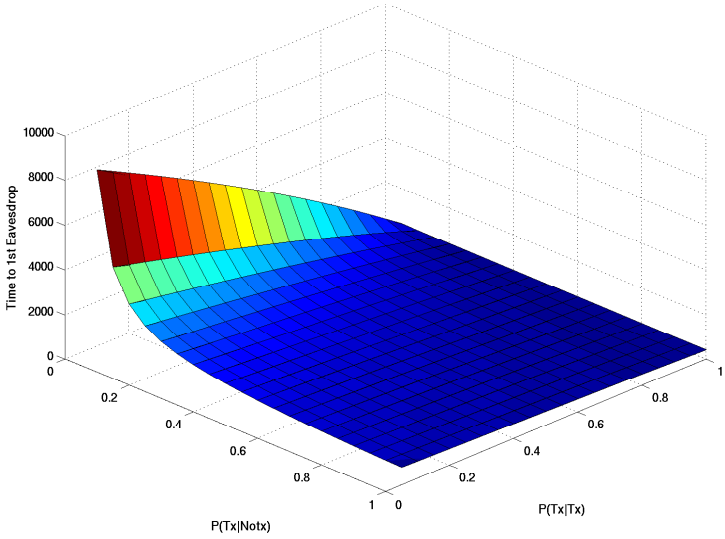


(a)

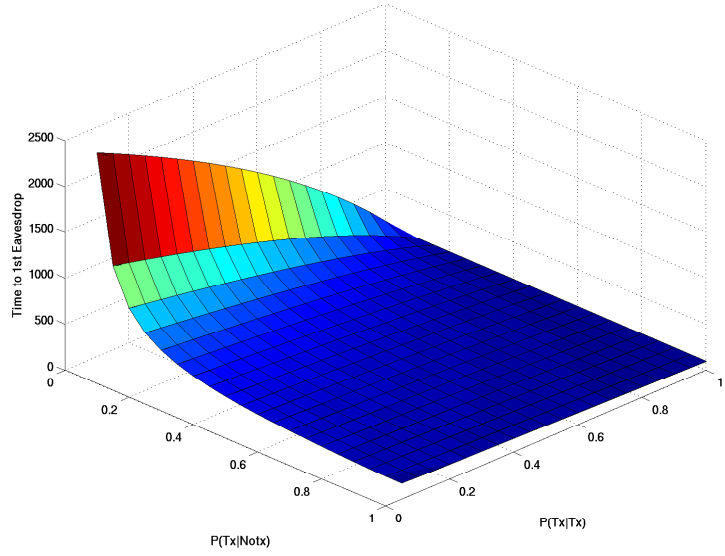


(b)

Figure 4.7: Perimeter Case: Time to Eavesdrop the First Packet



(a)



(b)

Figure 4.8: Corridor Case: Time to Eavesdrop the First Packet

```
P=? [ F<=T src1_coughtby_1=true ]
```

Listing 4.3: Property: Probability of Eavesdropping a packet within the next T seconds

Results are shown in Figures 4.9, 4.10, 4.11 and 4.12: $P(Eavesdropped)$ is the probability of eavesdropping a packet, T is the time within which the probability is computed, it can assume values of $(205, 505, 1003)$. Specifically:

- Figure 4.9 reports the case where the patient is near the window and the attacker is going along the perimeter.
- Figure 4.10 reports the case where the patient is far from the window and the attacker is going along the perimeter.
- Figure 4.11 reports the case where the patient is near the window and the attacker is walking within the hospital corridors.
- Figure 4.12 reports the case where the patient is far from the window and the attacker is walking within the hospital corridors.

The following can be observed:

1. The probability of eavesdropping a packet grows with $P(Tx|Tx)$ exponentially as e^{kx}, with $k < 1$; this growth was expected but not with a so small value of k.
2. The probability of eavesdropping a packet grows with $P(Tx|NoTx)$ as $1 - e^{-kx}$ with $k > 5$, this growth as well was expected but with a smaller k.
3. Values of $P(Tx|Tx) < 0.4$ do not influence $P(Eavesdropped)$ that much (see the graphs in the first row of each figure).
4. The higher T the higher the $P(Eavesdropped)$, this is obvious, the more time the attacker has to eavesdrop the more likely the insulin pump transmits a message.
5. For given $P(Tx|NoTx)$, $P(Tx|Tx)$ does not have any significant impact on $P(Eavesdropped)$. This is the most interesting result as it gives an indication on how to modify the design of the system (e.g. by tuning transmission probabilities) to make it more secure to some extent.

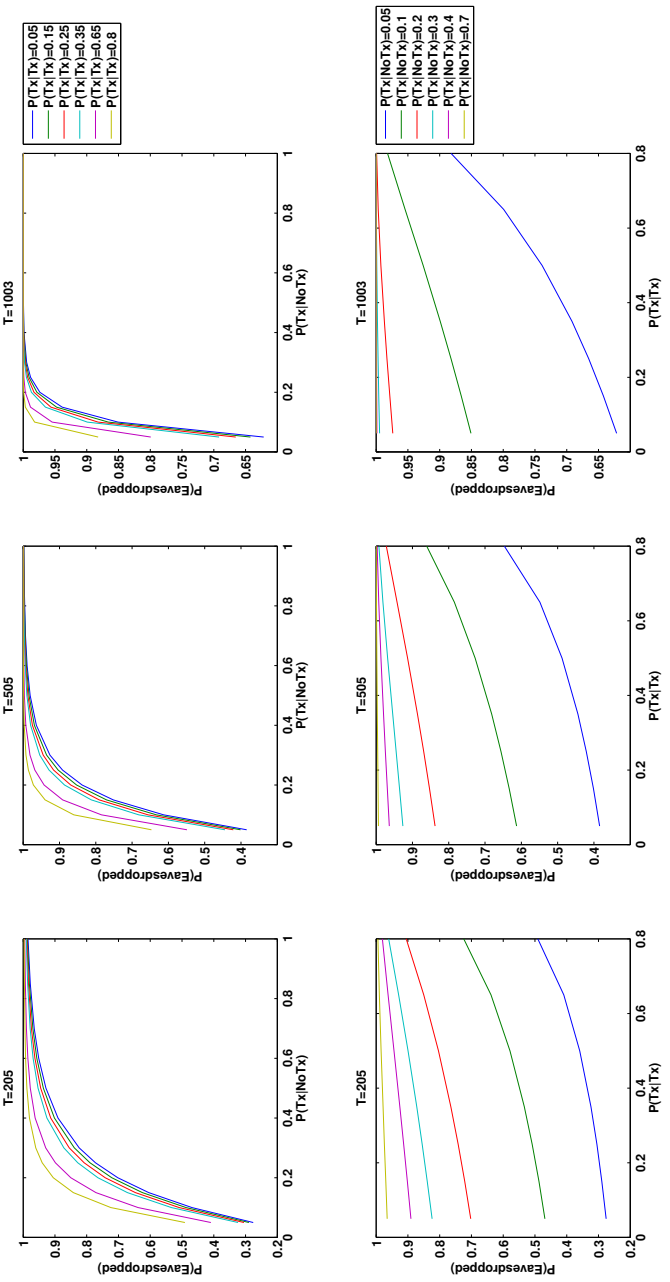


Figure 4.9: Eavesdropping Probabilities: Perimeter Case - Patient Near Window

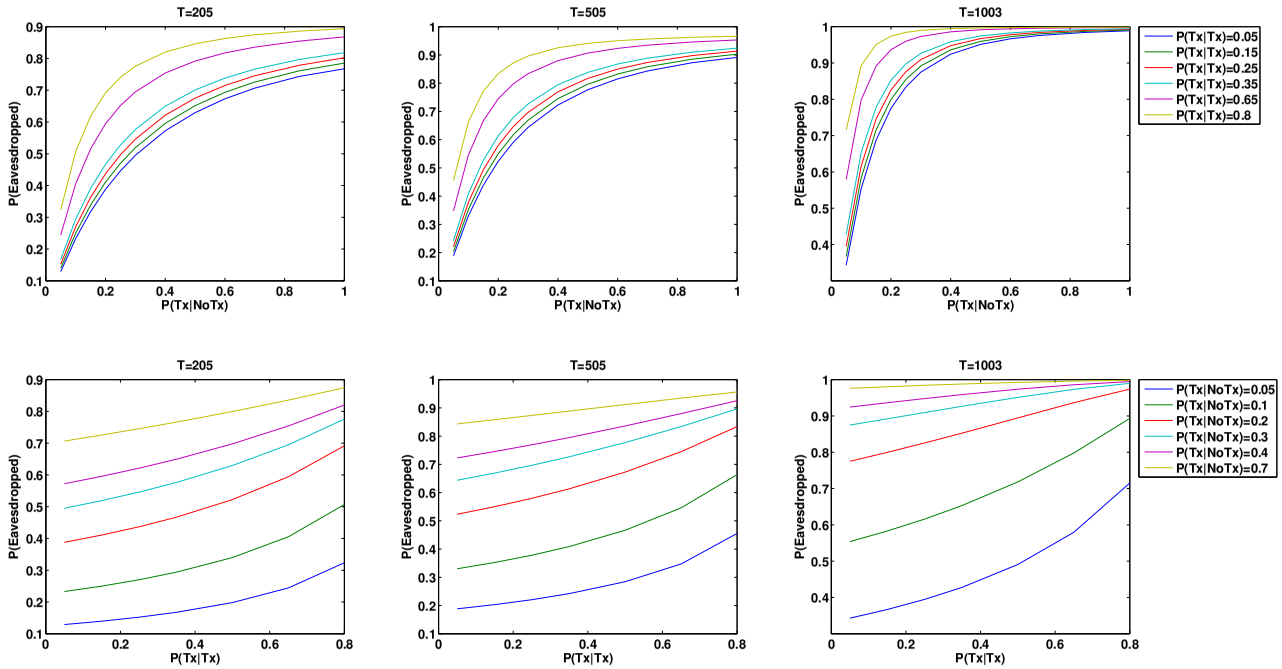


Figure 4.10: Eavesdropping Probabilities: Perimeter Case - Patient Far from Window

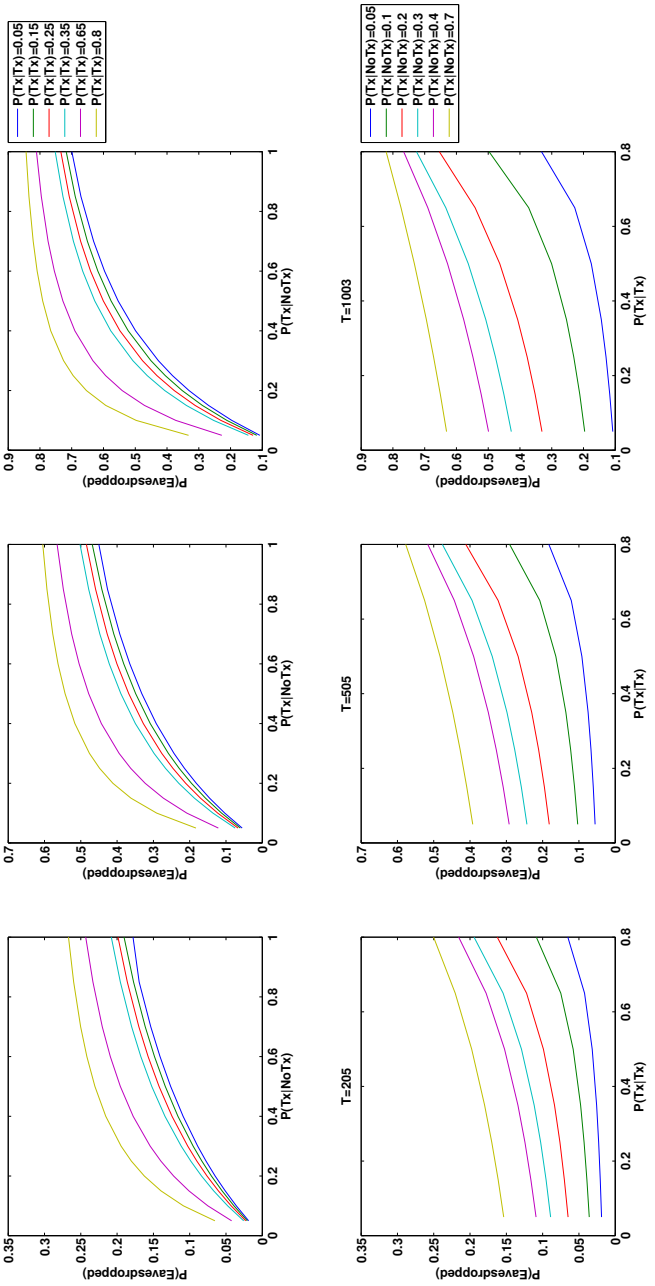


Figure 4.11: Eavesdropping Probabilities: Corridor Case - Patient Near Window

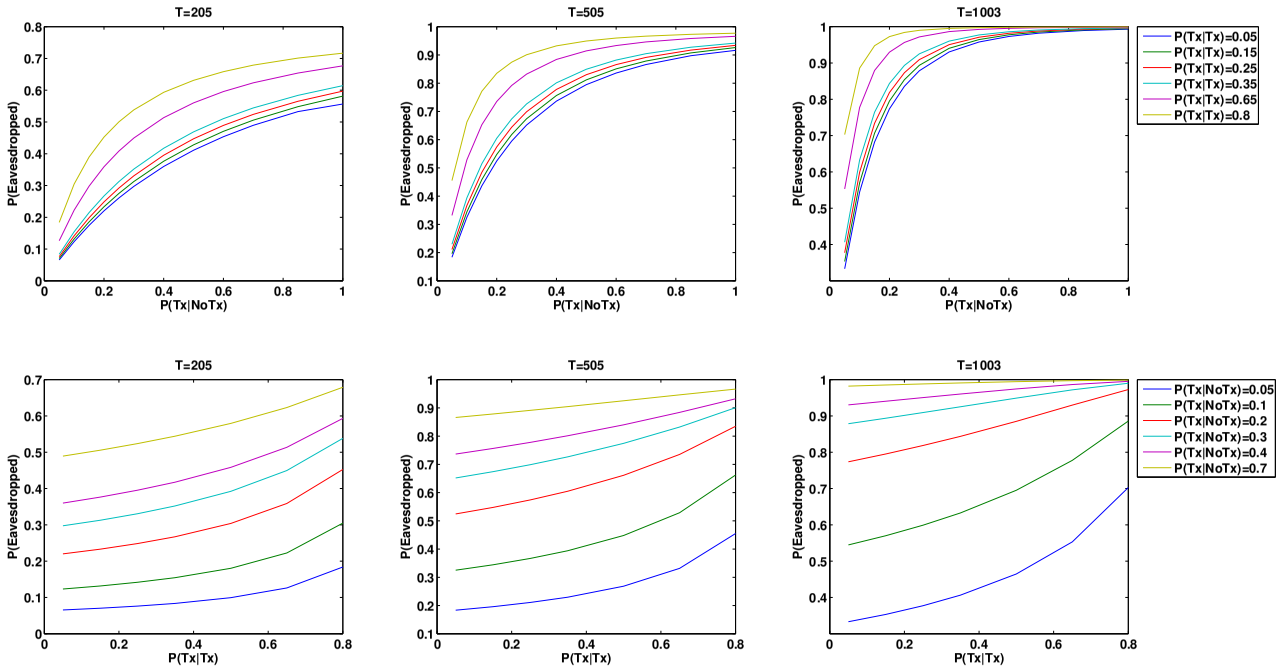


Figure 4.12: Eavesdropping Probabilities: Corridor Case - Patient far Window

4.2.3 Following the Target Device

In the previous part of this chapter we have investigated the eavesdropping of the first packet of the insulin pump, the main problem for the attacker was to find a target whose location was approximately known; from the results we have found out that it takes time for the attacker to find the insulin pump, mainly because its transmission range is very limited. But what happens after the attacker has found the patient? Clearly she would not continue to wander around but she would try to follow the patient, so the question arises: how does the model change in this case?

In order to investigate this scenario a new model has been created and implemented. This model is simpler: one of the two (the attacker) is kept fixed while the other (the patient) can freely move. The main difference is that the attacker is following the patient so that this one is always in range of the attacker. Hence the model is defined by the following parameters:

- $x = 1 : 5, y = 1 : 5$, the maximum distance between the patient and the attacker is 2 since the attacker is fixed in the middle the unit space is still $5m$.
- One mobile source, the patient.
- One static eavesdropper, the attacker, positioned in the middle of the space.
- $step_x$ and $step_y$ are both set to 1.
- The eavesdropping probabilities are set to $[0.95, 0.85, 0.1]$.
- $P(Tx|NoTx)$ and $P(Tx|Tx)$ are kept as variables.

The size of the model is only 100 states and 1064 transitions. Results shows that the time to eavesdrop packets still strongly depends on $P(Tx|NoTx)$. Figures 4.13 and 4.14 shows the probability of eavesdropping a packet with respect to time. It is possible to notice that when $P(Tx|NoTx)$ grows and $P(Tx|Tx)$ is fixed, the probability of eavesdropping grows as well noticeably, while when the opposite applies, still the probability grows with $P(Tx|Tx)$ but less noticeably.

Simulation Time

In this small section some figures about the simulation time are reported. The time needed to perform a simulation depends on the size of the model and on the type of property checked. Furthermore the simulations were parametrised on several values of transmission and retransmission probability, thus increasing the time.

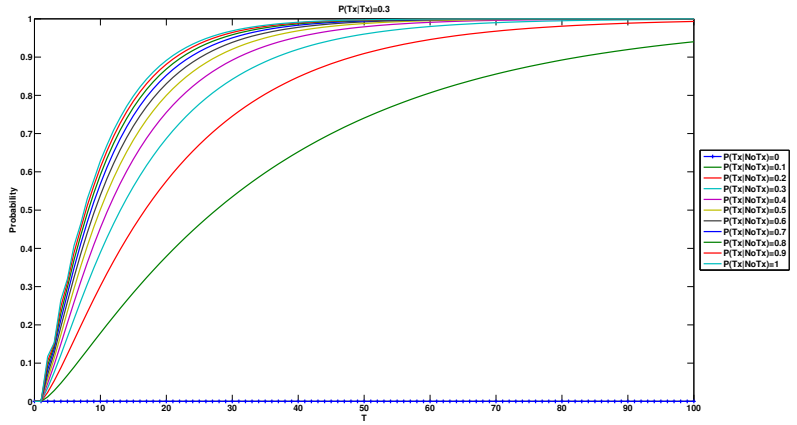


Figure 4.13: Probability of Eavesdropping a Packet with Respect to Time, P(Tx|Tx) Fixed

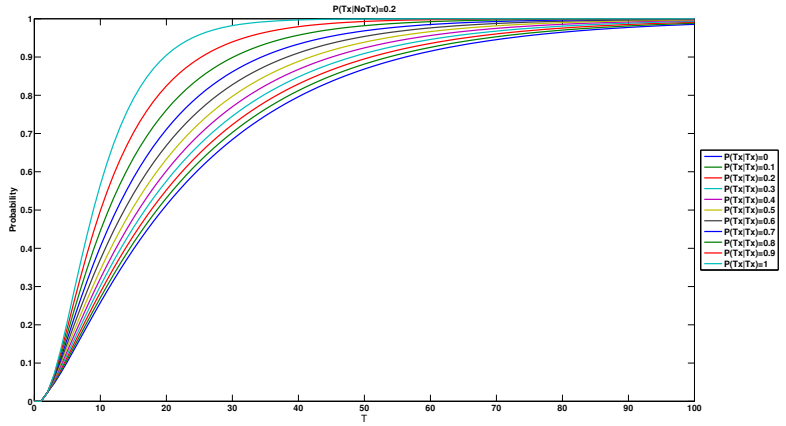


Figure 4.14: Probability of Eavesdropping a Packet with Respect to Time, P(Tx|NoTx) Fixed

Simulation	Minimal Time	Maximum Time
Property in Listing 4.2 Section 4.2.2	5 hours	30 hours
Property in Listing 4.3 Section 4.2.2	8 hours	50 hours
Following Target Section 4.2.3	5 minutes	20 minutes

Table 4.3: Simulation Time for Eavesdropping Mobility Model

Table 4.3 shows the simulation time for the eavesdropping mobility model presented in this section. The first two rows refer to the results in Section 4.2.2 while the third to the ones in Section 4.2.3.

4.3 Probability Extraction for Other Interventions

In the previous section we have seen how to extract probabilities for *Eavesdropping*. But how do we do that for other interventions, furthermore is it always the case where a general model can be defined and then tailored to the specifications of the system? The author will try to answer these questions in this section, unfortunately the answer is that it is not always possible to apply a general model, nevertheless it could very well be the case where the analyst chooses reasonable probabilities simply by appealing to his experience.

Most of the interventions defined in Chapter 3 are composed of atomic actions; as we already know probability extraction is the process of finding the probabilities of these actions being performed by the attacker. Interventions share several of these actions: as an example localisation is one of the first steps for most interventions. This makes the whole process of extracting probabilities shorter as once the probability of an action is extracted this can indeed be used in all interventions. The writer will now list all the actions within the interventions modules as defined in Chapter 3 and point out a way to extract the relative probabilities. Sometimes extracting a probability can be defined to some extent regardless the technology, as the process can be generalised from system to system, other times a line of action can be pointed out, but is the analyst that ultimately has to define its own way to extract the probability.

Device Location - This action is is part of all interventions but *Eavesdropping* and *Device Injection*. The probability of locating a device can be computed using the eavesdropping model defined above. Depending on the system, localisation can be done for example by triangulating the signal; to do this the attacker needs to eavesdrop three packets, and this probability can be easily computed with the mobility model above. Other ways of locating a device can be devised, most of them would probably rely on eavesdropping, hence the mobility model can still be used.

Device Crashing - This action is part of *Destruction*, specifically $A_{0,1} \rightarrow A_{1,S}$ (cf. Figure 3.6). The probability should be computed by the analyst depending on the underling technology (e.g. . In the case where the device does not have any specific design to prevent destruction, this probability can be set to one.

Remote Destruction - This action completes *Destruction* on the second branch of the intervention (cf. Figure 3.6, $A_{0,1} \rightarrow A_{2,S}, A_{3,S}$). With this the attacker is able to remotely destroy a device without localisation and it really depends on the system and on the vulnerabilities it has with respect to remote destruction (e.g. electromagnetic pulses), hence the analyst has to compute it.

Infer Data from Environment - This action is on the first branch of *Data Knowledge* (cf. Figure 3.8, $A_{0,1} \rightarrow A_{1,S}$). With this the attacker knows data on the memory of a device by inferring it from the environment (e.g. a temperature sensor). Performing this action should not be that difficult (e.g. measure the temperature of a place), but it is up to the analyst to choose that. Other memory values the attacker could be interested in are logic values such as routing tables, number of nearby devices etc. This kind of information can be obtained from the attacker by eavesdropping incoming transmissions, hence the eavesdropping model can be used in this case.

Device Physical Access - This action is part of *Data Knowledge*, *Full Data Modification*, *Reprogramming*, *Energy Reduction* and *Energy Control*. It represents the physical access to a device, it really depends on the device whether it has tamper proof mechanisms or not. If that is the case it is possible to extract the probabilities by looking at the specifications of the tamper proof mechanism, specifically on the statistics of tests performed by the constructor, if these are available. If the device is not tamper proof then the probability can be reasonably set to one, this is however up to the analyst.

Memory Read Access - This action is part of *Data Knowledge* and *Full Data Modification*, it is the physical access to the memory of a device. Similarly to the previous action, this could be inhibited or blocked by tamper proof mechanisms. Furthermore it could be the case where the access to the memory is secured with a password or a key (e.g. the Atmel CryptoMemory [1]). Such devices are hard to break in but a skillful attacker could hack it (e.g. see the DPA to Atmel CryptoMemory in [7]). Extracting probabilities for this action requires the analyst to rely on his experience, again if there are no tamper proof mechanisms this probability could be set to one.

Feed Value Physically - This action is part of *Disturb/Partial Data Modification* (cf. Figure 3.9, $A_{0,1} \rightarrow A_{1,S}$), it is the modification of a value by forging it physically (e.g. a sensor reading). For this action there is no general way to extract the probability hence the analyst needs to assess it case by case. Nevertheless in principle, when is the case of forging sensor data, the analyst could use the data-sheet of the device, to assess the timing responses of the sensor and its precision. Once this probability is computed, the successful feeding (see Figure 3.9, $A_{2,S}$) probability could be set to one or to other values depending on the system implementation (e.g. choose a value by majority voting with three consecutive readings).

Fake Traffic Injection - This action can be seen as complementary to eavesdropping: the creation and transmission of arbitrary data over the network. Indeed it requires the knowledge of data formats and how the system works. It is part of *Disturb/Partial Data Modification* and *Energy Reduction*. The analyst can compute it by taking into account the sensitivity of the target receiver, the distance between transceiver and receiver and the maximum transmission power. It is exactly as eavesdropping, but it does not need a complex model such as the mobility model presented above, since we can assume that the attacker already knows that a device is out there listening, however the mobility model can be used by the analyst if needed; in this case the probability of eavesdropping, $P(Eavesdropping)$ becomes the probability of successful injection, $P(Injection)$, the overall probability of transmission can be set to one (i.e. the attacker tries to inject packet continuously). Once the probability of this action is computed, the analyst has to assign a probability to $A_{0,2} \rightarrow A_{3,S}$ (i.e. the device accepts the fake data, cf. Figure 3.9). Similarly to "Feed Value Physically", this probability could be set to one or to other values depending on the system implementation.

Memory Write Access - This action is part of *Reprogramming* and *Full Data Modification*, for which is a key action as it is impossible to modify data on a device without it (cf. Figure 3.9, $A_{1,S} \rightarrow A_{3,S}\ A_{2,S} \rightarrow A_{4,S}$). It is complementary to "Memory Read Access" as it gives the attacker the power to modify the contents of a device memory. There are two different type of memories: data and program. Accessing one could be easier than accessing the other. Both memories could be protected with tamper proof mechanisms or by cryptography and crypto chip. Furthermore some parts of the memory could be hard-coded into the device (e.g. by hardware design), hence impossible to change. About the program memory, some devices have special interfaces that permit complete access to the program memory (e.g. JTAG interface [2]), therefore sometimes access to program memory could be easier than accessing Data memory. The remarks for the analyst are similar to the case of "Memory Read Access": the analyst has to rely on his experience, if there are no tamper proof mechanisms or if some special interface is present on the device, this probability could be set to one.

Remote Reprogramming - This action belongs to the second branch of *Reprogramming* (cf. Figure 3.11). It is the remote reprogramming of a device, this could be done by malware spreading or by exploiting legitimate mechanisms such as remote firmware upgrades (see [58]). This probability has to be assessed case by case as it is closely tied up with the underlying technology. What the analyst could do is simply to put this probability either to zero or one depending on the attacker strength he is considering for the analysis.

Inject Device - This is the only and fundamental action for *Device Injection*. the success of this action depends uniquely on the technology used and on the system policies. The intervention, *Device Injection*, could be supported by *Destruction*, (cf. Table 2.1), hence the analyst could take this into account when computing the corresponding probability: the destruction of a device could ease greatly the injection of a new device in the system.

Remote Energy Reduction - This is part of *Energy Reduction* (cf. Figure 3.13,$A_{2,S}$). It is the action of reducing the energy of a device without having access to the device. Basically this can happen by inducing the device to consume its energy by making him respond to network traffic for example. This action needs the attacker to be able to inject some traffic that the device reacts upon. In the extraction of this probability the analyst could simply compute it as with "Fake Traffic Injection" combining it with the energy control mechanisms that a device might have which might be transmit low rate data to save

energy. Clearly this action has to be assessed case by case.

Physically Force Data Transmission - This action is part of *Energy Reduction* (cf. Figure 3.13,$A_{4,S}$). It stimulates the transmission of data by exploiting the device nature without tampering with it (e.g. changing the temperature or moving the sensor). Clearly it can be performed only if the device is located. In extracting this probability the analyst should take into account the device specifications, specifically the time the sensor needs to respond to the action (a sensor reading could occur every fixed period of time), how is it designed to react to the stimuli the attacker might perform and other possibly more complex behaviours.

Energy Reduction by Tampering - This action is part of *Energy Reduction* (cf. Figure 3.13, $A_{3,S}$). It needs the attacker to locate the device and to be able to tamper with the device. As with all other actions that deals with tampering, tamper proof mechanisms have to be taken into account by the analyst when extracting probabilities for this action. The actions depends also on the type of energy source the device has, whether is a battery or something else, and on the power control unit. Performing this action on energy harvesting devices should be more complex, as the power control unit is designed to control multiple power sources hence follows some algorithms which prevents energy depletion in general.

Energy Control by Tampering - This action is part of *Energy Reduction* (cf. Figure 3.14, $A_{2,S}$ and $A_{3,S}$). It is very much similar to the previous apart from the fact that the energy can be also risen by the attacker and not only depleted. This could be done by taking in and out the battery at specific points in time, or by exerting some specific kind of control over the power control unit of the device. It is very specific to the system so the analyst ultimately has to deal with the problem of computing the transition probabilities.

Control Energy Source - The second branch of *Energy Control* (cf. Figure 3.14) define a state where the attacker controls the energy level of a device without tampering with it (i.e. without physically accessing the device). This can be done with energy harvesting units by controlling the energy source that the device harvests. It is indeed very specific to the kind of energy source and on how this energy is harvested. A general method to extract probabilities for each harvesting method could be indeed developed, but doing it within this thesis would not make much sense since it is very expensive in terms of time

and secondly because technologies change rapidly, and such an analysis would easily need a major reassessment in a few years. Still, when the analyst extract the probabilities for a specific energy harvesting method, he can then reuse it in the future without the need of computing them again.

This completes the list of actions for the interventions. Within each action the analyst, together with probabilities, has to account also for the time each action takes, so that he can input it in the model as a reward coefficient (remember the example in Chapter 3, Section 3.3.2, Figure 3.15).

	Analyst Experience	General Model	Tech Specific
Device Location		x	
Device Crashing			x
Remote Destruction			x
Infer Data from Environment		x	x
Device Physical Access			x
Memory Read Access	x		x
Feed Value Physically			x
Fake Traffic Injection		x	x
Memory Write Access			x
Remote Reprogramming	x		x
Inject Device			x
Remote Energy Reduction		x	
Physically Force Data Transmission			x
Energy Reduction by Tampering			x
Control Energy Source		x	x

Table 4.4: Actions and Probability Extraction Process

Table 4.4 shows the probability extraction method for each action defined above. The extraction can be based on a general model (e.g. device location), the analyst experience (e.g. memory read access) or be technology specific (like for example the energy source control). It is clear that most of them are all dependent on the technology, whether it is some tamper proof mechanism or a energy harvesting technology or a cryptographic chip.

In this chapter we have defined the process of probability extraction, built a model to compute eavesdropping probabilities for mobile and static station and pointed out a way to extract probabilities for every intervention. It should be clear to the reader that the analyst can disregard the computation of some probabilities, setting them to a neutral value (i.e. 1) or to a null one (i.e. 0). The analyst should do this when:

- For a professional or analytical reason, he is not interested in exploring specific states of the model: null value;
- The system under analysis does not have any implementation or behaviour or physical part which allow for some branches of the model (e.g. a device that cannot be reprogrammed remotely): all the transitions belonging to those branches assume a null value;
- He is not interested for professional or analytical reasons in the effect of a specific transition probability: the corresponding value is set to 1 (neutral value) so that in a path its contribution is neglected (i.e. multiplication by 1);
- He is not able to extract a probability for an action either because he does not have time or because he does not have the necessary data or documentation: neutral value.

From this remarks it is important to note that it is not mandatory to extract probabilities to perform an analysis. Even getting only the results about the eavesdropping probabilities with the mobility model can be of interest depending on the case.

4.4 Summary

In this chapter the author defined and explained the process of *Probability Extraction*. This is a critical process as it supplies the model with the quantitative values it needs. It is also a most complex process as the technologies that compose ubiquitous systems are various and of different natures; furthermore it is not always the case where the analyst has access to detailed system specifications. In the last part the author makes a classification of the possible ways for the analyst to extract probabilities. The chapter describes in details the process of extracting probabilities for eavesdropping: this is done by the development of a model which takes into account the movements of the target and the eavesdropper, along with the range the transmissions can be eavesdropped at.

Chapter 5

Analysis of two Systems

This chapter proposes the analysis of two systems, namely the Insulin Pump System, described in [43], and the FleGSens system, an area monitoring WSN, defined in [52]. The analysis is performed using the PRISM model checker and the model defined in the previous chapters.

In order to analyse a system the analyst has to perform three steps, these are:

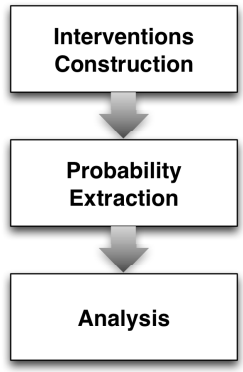


Figure 5.1: Analysis Process Flow Diagram

1. The construction of the interventions against each system, i.e. what are the active states or whether an intervention exists for the specific system.
2. Probability extraction, what are the probabilities assigned to each transition.
3. The analysis of the model using PRISM.

Figure 5.1 shows a flow diagram of the process.

The analyses proposed in this chapter are divided into four parts, one for each step plus a final discussion. As probability extraction has been thoroughly explained in Chapter 4,This chapter will focus on what numbers are used as input in the model and how they were chosen (remember Table 4.4 in Section 4.3).

5.1 The Insulin Pump System

The system has been already introduced in Chapter 2. It consists of five wirelessly interconnected devices (see Figure 5.2): an insulin pump, a remote control, a PDA, used for collecting data and reporting them to the patient or the doctor and to program some parameters of the system, a continuous glucose sensor, attached to the body, and a user glucose meter that can be used by the patient or the doctor to perform a measure of the glucose present in the blood stream, whenever needed.

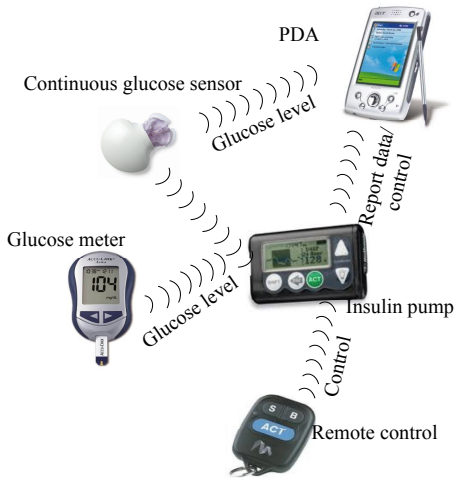


Figure 5.2: Insulin Pump System - Source [43]

The insulin pump is the main device, it communicates with all other devices and is the one which directly injects the insulin into the patient. The glucose sensor monitors the glucose level of the patient and the whole system relies on its readings. It sample the glucose level of the patient with a sampling frequency that can be defined by the doctor (in this thesis we assume that it gather a sample every 30 seconds). The remote control can be used to tell the insulin pump to inject a dose of insulin in the patient's body. It can be used by the patient to adjust his insulin dose whenever he feels like he needs it[1]. Figure 5.2 shows the communication existing between the devices.

5.1.1 Construction of Interventions

As the interventions defined in Chapter 3 are general for Ubiquitous Systems, they have to be constructed against the insulin pump system. Since the Insulin Pump is a highly localised device (i.e. the human body), and its range does not spread over 7-10 meters, the interventions' branches that relies on purely remote actions can be generally discarded. The interventions are therefore constructed as follows:

Destruction: The intervention only involves the first branch states: physical location $A_{0,1}$, and crashing $A_{1,S/F}$.

Eavesdropping: This intervention has several states $A_{1,S/F} \dots A_{n,S/F}$ depending on how many time/probabilities combinations the analyst wants to explore. This was already taken into account in the mobility model in Chapter 4. Section 5.1.2 explains how values are chosen from the model.

Data Knowledge: All states are active. The attacker could get knowledge of the values of the continuous glucose sensor for example by eavesdropping its transmissions, or she could get hold of a device and access its memory.

Disturb/Partial Data Modification: The intervention involves all states. The first branch expresses the fact that the attacker feeds the device with a false reading, in this case it could be that she feeds the glucose meter with a false sample. With respect to the second branch the attacker could forge legitimate command packets hence forcing the insulin pump into injecting an irregular dose (as the attack in [43]).

Full Data Modification: All states are active as the attacker could get hold of a device hence try to access its memory and modify it (remember this intervention needs physical access to the device, cf. Section 3.3).

[1] Diabetic patients are usually trained in self administration of insulin doses.

Reprogramming: As in the previous interventions all states are active as the attacker could access one of the devices (e.g. the PDA) and reprogram it both remotely, through malware or system update, and locally by reprogramming one of the devices such as for example the glucose meter or the remote control.

Device Injection: This intervention is active as well, as the attacker could substitute the remote control of a system with one programmed by herself, or she could also destroy the continuous glucose sensor and inject a new one programmed by herself.

Energy Reduction and Control: It does not make sense to talk about these last two interventions since energy management is not critical for the Insulin Pump System.

The model describes all the possible actions the attacker can take, even if these are very unlikely, complex or too risky to be carried out. The attacker for example could prefer to eavesdrop and inject packets from a certain distance instead of trying to steal a device and being caught in the process, hence the analyst might want to explore only the most likely threats instead of all of them.

5.1.2 Probability Extraction

This section discusses and defines, for the relevant actions, the values for the probabilities and the time rewards which are used for the analysis of the model in Section 5.1.3.

Device Localization - As device localization is the basic action it is only logical that it is the first probability to be extracted. To do this the mobility scenario defined in Chapter 4 is used. If we consider the average walking speed for a human as $4.5km/h = 1.25m/s$ and that the unit space is $5m$, then the time unit is $\Delta T = \frac{5m}{1.25m/s} = 4sec.$ The transmission of the Insulin Pump can be modelled as a Poisson process. In a Poisson process the probability of having k events in a time ΔT is given by:

$$P(N(\Delta T) = k) = \frac{(\lambda \Delta T)^k e^{-\lambda \Delta T}}{k!} \tag{5.1}$$

where λ is the transmission rate. The most active device in the insulin pump system is the continuous glucose sensor which transmits two measurements every

minute. As the mobility model is time discrete, the transmission probability has to be computed with respect to the time unit. Hence the probability of having at least one transmission in a time unit is:

$$P(N(\Delta T) > 0) = 1 - P(N(\Delta T) = 0) = 1 - (\lambda 4)^0 e^{-\lambda 4} \approxeq 0.125$$

$$with\ \lambda = 2/60\ and\ \Delta T = 4s.$$

By looking at the eavesdropping probabilities curves in Figures 4.9, 4.10, 4.11 and 4.12, for values of $P(Tx|NoTx) = 0.125$ the corresponding $P(Eavesdropped)$ values are:

- 0.04 to 0.4 for $T = 205$, approximately 14 minutes.
- 0.12 to 0.58 for $T = 505$, approximately 34 minutes.
- 0.2 to 0.85 for $T = 1003$, approximately 67 minutes.

These are the probabilities of eavesdropping the first packet. When the attacker does it, she knows she is within 7 meters from the victim, hence she just needs to eavesdrop few more packets (roughly a minute) to refine the location of the target.

These values apply for an attacker that follows the mobility model in Chapter 4, which models one mobile eavesdropper and one mobile station. Indeed it is possible to stage a stronger attacker in the model, for example an attacker that is able to place multiple eavesdropping devices in the environment, thus increasing the eavesdropping probability and decreasing the time to eavesdrop accordingly. The strongest attacker eavesdrops a packet after 30 seconds with probability 1.

Device Crashing - As for device crashing, it is up to the analyst to assess the probability and the time for the attacker to crash a device. In the case of the insulin pump it depends very much on which of the five devices she wants to crash, getting hold of the remote control it would probably be easier than the continuous glucose monitor or the insulin pump which are attached to the body, and the attacker might exploit the fact that in a hospital it is normal for patients to take off their insulin pump systems for reprogramming or maintenance. It is also a question of how much risk the attacker wants to take to carry out her action. In this analysis we will assume that the attacker wants to destroy the continuous glucose sensor since it is the most likely device to be targeted as it is the most active one. Furthermore if the attacker wants to send false glucose measurements, the continuous glucose sensor has to be taken off, otherwise the

insulin pump could discard the fake measurement in favour of the real one. Here we set the probability of destroying the device to 1 with a relatively long time, 10 minutes (this value is empirical but reasonably realistic). The attacker might also like to destroy the insulin pump, but here we discard this case since the aim of this analysis is to evaluate the security and integrity of the insulin pump itself with respect to the surrounding threats, moreover as the insulin pump is the core of the system, it must be physically well protected, hence crashing it very unlikely.

Inject Device - Crashing the continuous glucose sensor can be used to support device injection; this can be modelled as a non-deterministic choice in modelling *Device Injection*. This can be done simply by duplicating the PRISM module and assigning different probabilities and time for execution to P_1 (cf. Figure 3.12). In the case of injection being supported by destruction, we set the probability to 1 with a value for the time reward comparable with the interval between measurements (i.e. 30 seconds). In the opposite case (injection not supported), the overall time is shorter (i.e. the attacker does not have to destroy a device), but the probability of success is lower. In this work it is set to 0.5 (it is assumed here that the insulin pump decides either for the legitimate sensor or the fake one). Unfortunately the technical specifications of the insulin pump could not be retrieved the only data at disposal are the ones in [43]. With them a more precise probability for device injection could be computed.

Infer Data from Environment - The only data that the insulin pump uses are glucose samples. The attacker can get hold of this data by eavesdropping transmissions from the continuous glucose sensor. As the packets are not encrypted, all the attacker needs is just to catch a transmission. To compute this probability, given that the device is located, the eavesdropping mobility model can be used: this is the case where the device is followed by the attacker (cf. Section 4.2.3). Specifically the analyst can use the data in Figure 4.13: for $P(Tx|NoTx) \approxeq 0.125$ the average time to eavesdrop a packet is $T = 50\, Time\, Units$, approximately 3-4 minutes, with a probability of $\approxeq 0.9$.

Memory Read Access - Data about the glucose samples can be acquired by the attacker also by accessing the memory of a device. In the insulin pump system the PDA is the only device that stores data. Unfortunately specific details about PDA (e.g. operating system, storage method) are not available; anyway it is very likely for this data to be unprotected, as the transmission is unprotected as well. For this reason in this analysis we assume that for the attacker it should not be a problem to extract the data from the PDA once she

has it in her hands. Moreover if data are stored on an external memory card, accessing them could even be easier.

Device Physical Access - As there are no technical specifications it is not possible to assess this probability, hence we will disregard it (i.e. set it to 1). Nevertheless it is possible to make some observations: the attacker might want to access the insulin pump itself to reprogram it for a specific purpose or decide instead for the continuous glucose sensor to tamper with its detection mechanisms. The remote control could also be sabotaged by the attacker. Device physical access concerns with the PDA as well, as there could be mechanisms in place to inhibit access to the device (such as secure data wipe if password is not correct and similar). All these actions indeed need the attacker first to get hold of the device (as in device crashing), this can be taken into consideration when inputting the probability and the time reward. In this analysis *Device Physical Access* comprises getting hold of the device and accessing it.

Fake Traffic Injection - Computing this probability is similar to eavesdropping and it is as much as important, since the attacker may forge a transmission to force the insulin pump to inject an irregular dose of insulin. The range for injecting a packet is $20m$ with a $200mW$ ($23dBm$) transmitter (as [43] states). Hence depending on the scenario a model similar to eavesdropping could be produced. It is also possible to assume that the attacker is able to get within the range of the insulin pump once she has identified the target. As there is always a small possibility of failing even if the insulin pump is well within range, the probability here is set to 0.97 (this is an average value chosen considering data in Figure 4.2). The time needed is simply the time for the transmission (i.e. the time unit).

Reprogramming - Reprogramming could be performed on every device. The glucose meter, the sensor, the insulin pump and the remote control need specific knowledge about the hardware hence also specific skills from the attacker. As for *Device Physical Access*, we do not have any technical specification therefore it is not possible to assess whether reprogramming is possible and how much time it takes. The PDA could also be targeted from the attacker, and since it has a general purpose OS, even a weak attacker should have the necessary skills to reprogram it for example through a malware remotely or locally by installing a malicious application directly on the device. As we do not have any specific technical specification we disregard this intervention here.

The process of probability extraction for the insulin pump ends here. This

could be improved if a specific technical documentation could be produced. Unfortunately this is not available mostly for security reasons: the authors of [43] did not provide the producer nor the model of the insulin pump as they would have clearly given evil people the instruction to attack the system, hence endangering actual patients lives.

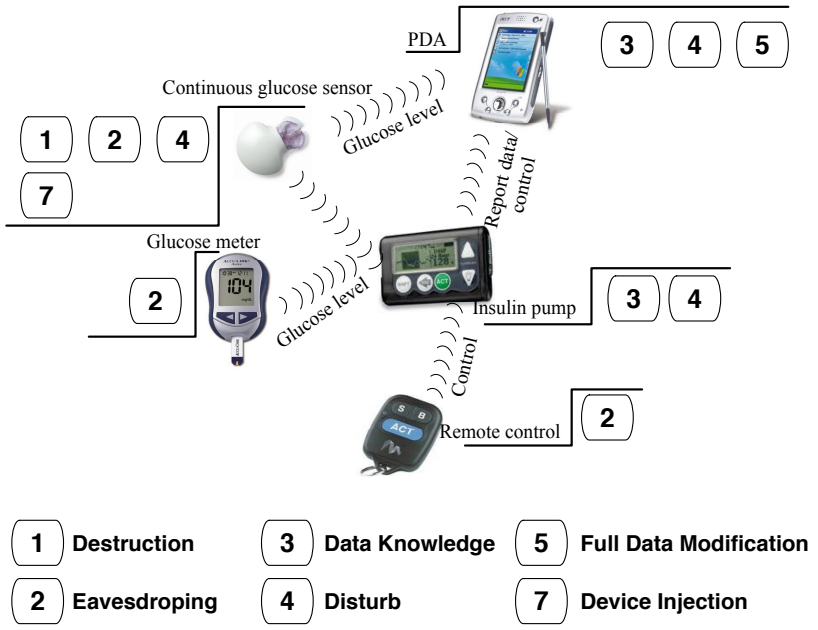


Figure 5.3: Insulin Pump Interventions Diagram

5.1.3 Analysis of the Model

Now that the probabilities have been decided, along with the time needed to perform the corresponding action, the analysis of the model with PRISM can be carried out. In Section 5.1.1 the author constructed the interventions for the insulin pump. Specifically, with respect to interventions defined in Chapter 3, some interventions have been discarded, others have been limited by making some states inactive. Remember that a state can be defined inactive for three reasons:

- The system under analysis does not express the behaviour the state represents.

- The analyst (which in this case is played by the writer) does not think it was worth investigating these states.
- The lack of technical specifications does not allow for probability extraction for the transitions bringing to those states.

The interventions modelled here are *Destruction* (cf. Figure 5.4), *Data Knowledge* (cf. Figure 5.6), *Disturb/Partial Data Modification* (cf. Figure 5.8), *Full Data Modification* (cf. Figure 5.10) and *Device Injection* (cf. Figure 5.12); inactive states are greyed out. Eavesdropping was largely discussed earlier, probabilities extracted from it are widely used throughout all interventions[2] together with the corresponding time rewards.

Figure 5.3 shows to which device each intervention can be applied to. Specifically *Eavesdropping* can be applied wherever a transmission occurs, hence at the glucose sensor, meter and the remote controller. In this analysis *Destruction* is applied only to the glucose sensor since is the device responsible for monitoring the patient glucose level and its destruction can support *Device Injection*. *Data Knowledge* can be applied to the insulin pump or to the PDA. With *Disturb/Partial Data Modification* the attacker forges a fake transmission from the glucose sensor hence targeting the receivers of such transmission: the insulin pump and the PDA. Finally *Full Data Modification* targets the PDA while *Device Injection* the glucose sensor.

Destruction - From the process of probability extraction, the transition probability $INIT \rightarrow A_{0,1}$ is set to 0.58 with a time reward of 505 time units. This is taken from the probability extraction of device location on page 95. The transition probability was chosen to be the worst value of the second line ("0.12 to 0.58 for $T = 505$, approximately 34 minutes."). This does not rule out the other values as there is a most important feature of this model (which often occurs in Markov chains models). We demonstrate this feature by showing it: when the model is in state INIT there is a probability of 0.58 to go to $A_{0,1}$ (cf. Figure 5.4), or 0.42 to stay in INIT. When you stay in INIT at the next step the model has the same choice again.

Going on, the probability of be in state $A_{0,1}$ (i.e. the attacker locates the device) is $P = 0.58$ at $T = 505$, $P = 0.58 + (0.42 \times 0.58)$ at $T = 1010$, $P = 0.58 + (0.42 \times 0.58) + (0.42 \times 0.42 \times 0.58)$ at $T = 1515 \ldots$ The results are consistent with those at page 95: for example $0.58 + 0.42 \times 0.58 \approxeq 0.83$ for $T = 1010$ which is approximately 0.85 for $T = 1003$. PRISM obviously takes this into account while checking the model.

[2]Remember that localisation which makes use of eavesdropping is largely used in other interventions.

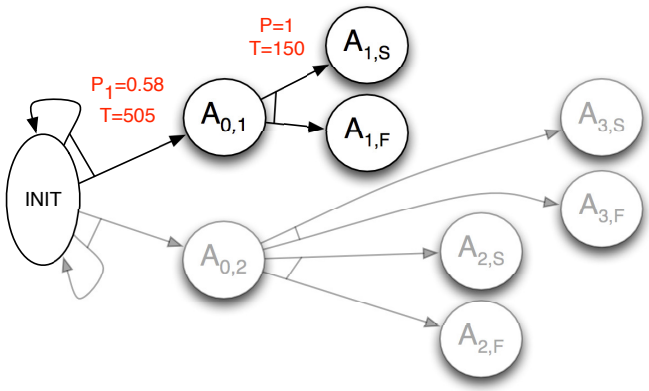


Figure 5.4: Insulin Pump: Destruction Intervention

In this analysis we are interested in the average time needed to achieve the intervention and on the relative probability. The reward structure that keeps track of the time is shown in Listing 5.1.

```
rewards "time"
    [locating] true:505;
    [crashing] true:150;
endrewards


// Property for model checking

Rmax=? [F A_d=2]
// A_d=2 corresponds to the
// successful destruction of the device
```

Listing 5.1: Insulin Pump: Time Rewards for Destruction

The expected time to destroy a device, which is the time needed to locate the device and destroy it, is $1020.70\,TU \approxeq 68\,min$. Figure 5.5 shows the probability of destroying the device as a function of time T (expressed in Time Units). As expected this probability grows with time, this because the more time passes the more likely it is for the attacker to achieve its goal.

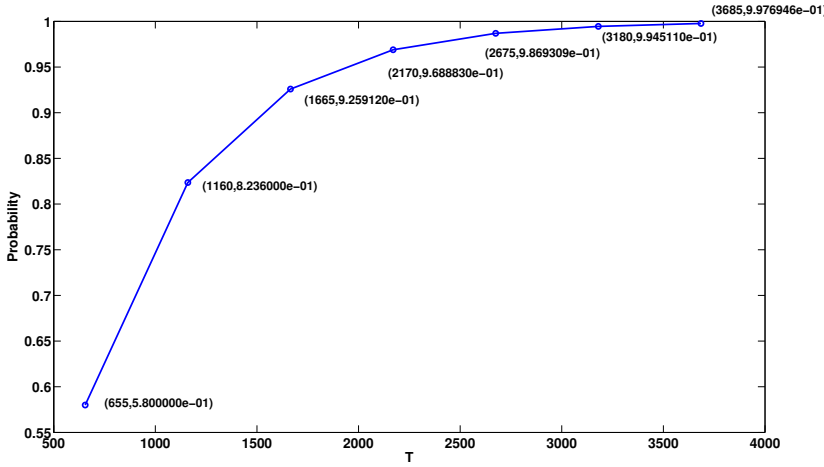


Figure 5.5: Insulin Pump: Success Probability for Device Destruction

Data Knowledge This intervention, differently from the one above, has two active branches which originate from state A_0 (cf. Figure 5.6). The corresponding reward structure is shown in Listing 5.2.

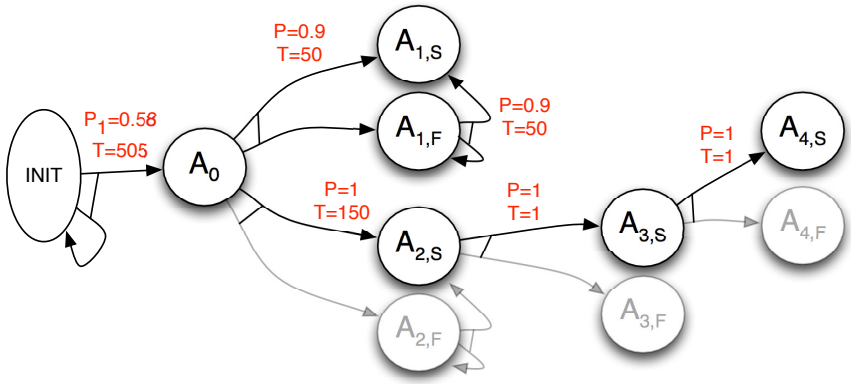


Figure 5.6: Insulin Pump: Data Knowledge Intervention

Specifically there is only one reward structure which keeps track of the time for all branches. Depending on the property the analyst checks, the PRISM model checker computes the time related to the branch the property refers to.

```
rewards "time"
    [localize] true: 505;
    [environment_read] true: 50;
    [access] true: 150;
    [readout] true: 1;
    [success] true: 1;
endrewards

//Property for time to reach state A1,S (A_dk=2)
Rmax=? [F A_dk=2]

//Property for time to reach state A4,S (A_dk=8)
Rmax=? [F A_dk=8]
```

Listing 5.2: Insulin Pump: Time Rewards for Data Knowledge

The expected time, computed by PRISM, to eavesdrop a packet and gather a reading from the glucose sensor (i.e. time to reach state $A_{1,S}$) is $926.24\,TU \approxeq$

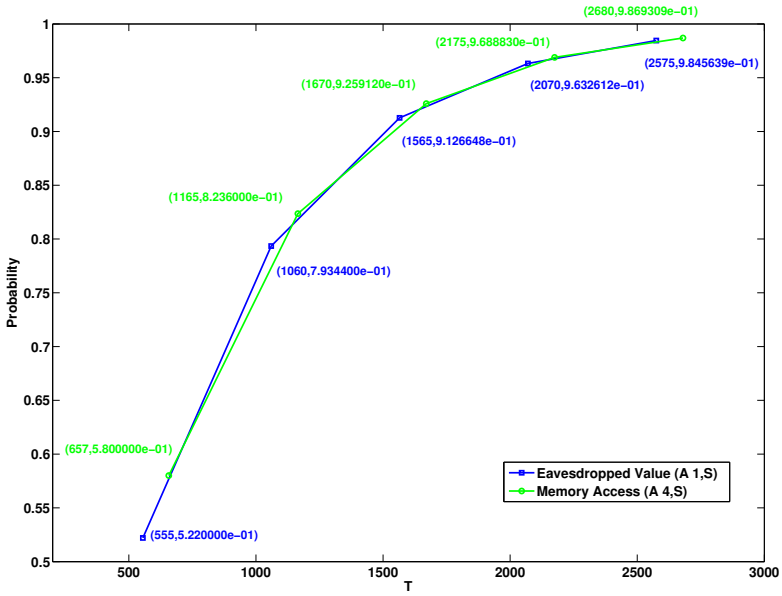


Figure 5.7: Insulin Pump: Success Probability for Data Knowledge

62 min, while the time to physically get access to the memory of a device (e.g. by getting hold of the PDA) is 1022.69 $TU \approxeq 68min$.

Figure 5.7 shows the probability as a function of time T (expressed in Time Units) of achieving data knowledge by reaching states $A_{1,S}$ (eavesdropping of the value) and $A_{4,S}$ (accessing the device memory). The two curves are comparable, this is because locating the device is the most time consuming action. Since the expected time is similar, the attacker can choose the action that carries the smallest risk (i.e. eavesdrop the packet). Indeed getting access to the device memory gives more information, hence it depends ultimately on which are the attacker's goals. An important observation is that the two actions are comparable as the transmission is given in clear (i.e. is not encrypted). If confidentiality were implemented, then the transition probabilities and the time rewards would change greatly, and accessing the device would most likely be the easiest way to achieve *Data Knowledge*,even though it is more risky.

Disturb/Partial Data Modification This intervention has five inactive states. These are the ones that express the behaviour of feeding a false value through the environment. As stated before this does not make much sense within the insulin pump system as modifying the glucose value detected from the sensor would mean to intervene on the patient and, whereas this is indeed possible, it is not possible to extract a general probability for this event.

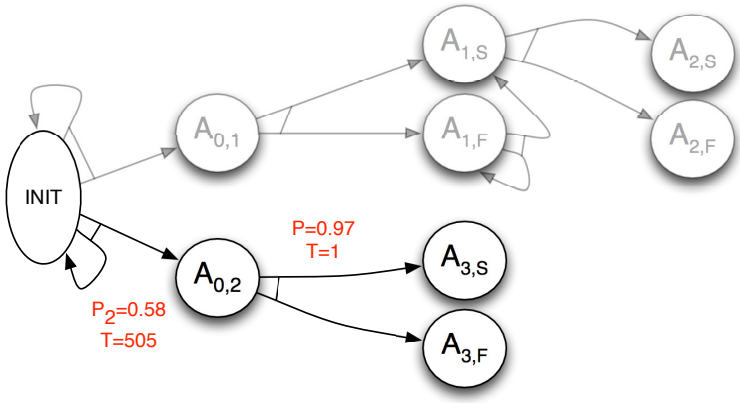


Figure 5.8: Insulin Pump Disturb/Partial Data Modification Intervention

To carry out this intervention the attacker has to forge a false transmission containing a false reading. This behaviour is expressed by the second branch (see Figure 5.8). As the insulin pump is a highly localised system the attacker has

first to locate it, the transition probability and the reward time of INIT$\rightarrow A_{0,1}$ are set at 0.58 and 505 TU. Then for $A_{0,2} \rightarrow A_{2,S}$, this is set to 1 (the time to send the transmission) and the probability to 0.97 (cf. "Fake Traffic Injection" page 97). It is important to note that in this case there is no "retry" transition $A_{2,F} \nrightarrow A_{2,S}$ since there is no way for the attacker to know whether the injected packet was accepted by the system.

```
rewards "time"
    [localize] true:505;
    [inject] true:1;
endrewards


// Property for model checking

Rmax=? [F A_dist=7]
// A_dist=7 corresponds to the
// successful injection of the forced transmission
```

Listing 5.3: Insulin Pump: Time Rewards for Disturb/Partial Data Modification

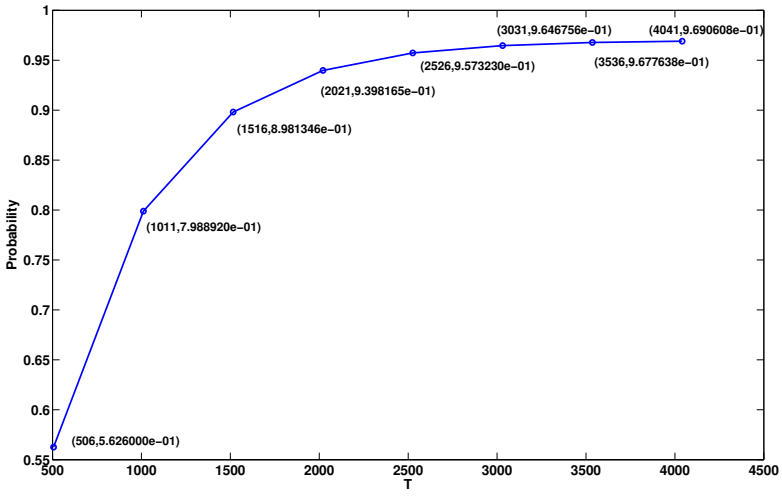


Figure 5.9: Insulin Pump: Success Probability for Disturb/Partial Data Modification

Listing 5.3 shows the reward structure for the time needed to accomplish the intervention. Figure 5.9 shows the probability for successfully injecting a false

transmission as a function of time T (expressed in Time Units). As with all other interventions whose branches are based on localisation, the success probability grows with time. Unlike other interventions the probability goes asymptotically to 0.97; this is the maximum probability for achieving the intervention as $P(A_{0,2} \rightarrow A_{3,S}) = 0.97$ and $A_{3,S}$ is an absorbing state (i.e. there is no retry transition from the corresponding fail state).

Full Data Modification With this intervention we consider the fact that the attacker wants to get hold of the PDA and to modify its memory. The PDA is the only device in the system that stores data about the glucose readings and the system status hence it is the only device this intervention applies to. Full Data Modification is divided into two branches, one that enables the attacker with read/write access to the device, the other which enables write access only. The two branches originates from state $A_{1,S}$ (cf. Figure 5.10) that represents the fact that the attacker has located the device and has gathered physical access to it. In this case the branch with write access only is inactive as accessing a common PDA memory usually means that it can be both read and written at the same time, and there are no indications that mechanisms to prevent this are in place. As a marginal note, the write only branch would apply only to cases where the memory can only be written (e.g. a hardware register).

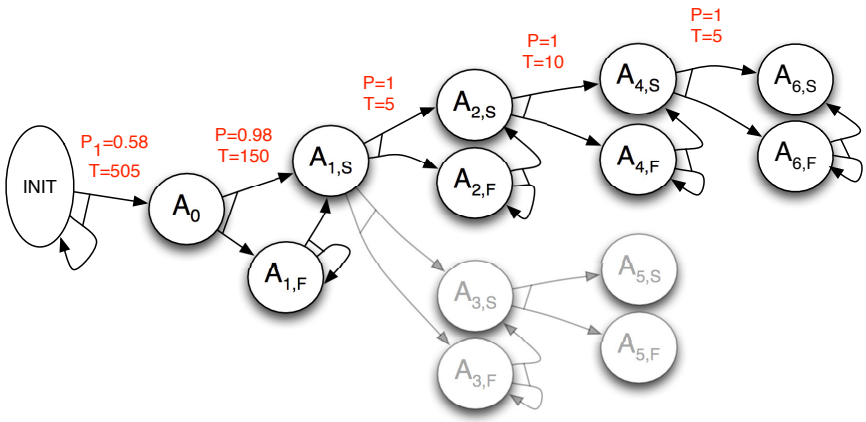


Figure 5.10: Insulin Pump: Full Data Modification Intervention

The corresponding reward structure is shown in Listing 5.4. The time to access the device is set at $150\,TU \approxeq 10\,min$ which is the same as the time for device crashing (cf. page 96). The probability is set to an arbitrary high value 0.98 (and not to 1) to express the possibility for the attacker to fail in getting physical

```
rewards "time"
    [locating] true:505;
    [access] true :150;
    [read_access] true :5;
    [write_access] true :10;
    [success] true :5;
endrewards

// Property for model checking

Rmax=? [F A_fdm=8]
// A_fdm=8 corresponds to the
// successful injection of the forced transmission
```

Listing 5.4: Insulin Pump: Time Rewards for Full Data Modification

access to the device (i.e. fail in stealing it from the patient or the doctor). The attacker may try again to get the device hence the transition $A_{1,F} \rightarrow A_{1,S}$ is active with the same time reward and probability as $A_0 \rightarrow A_{1,S}$.

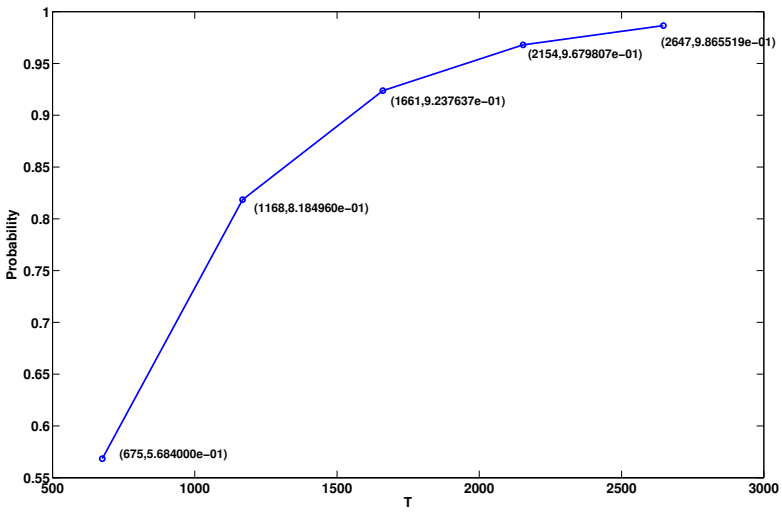


Figure 5.11: Insulin Pump: Success Probability for Full Data Modification

Figure 5.11 shows the probability of achieving *Full Data Modification* as a function of time T (expressed in Time Units). The values are comparable with the

previous interventions. This is of course expected as this intervention is based on localisation as well as the previous ones.

Device Injection With *Device Injection* in the case of the insulin pump system, we consider the case where the attacker wants to inject a continuous glucose sensor into the system. This device is a malicious device programmed by the attacker to send to the Insulin Pump a glucose value defined by the attacker herself.

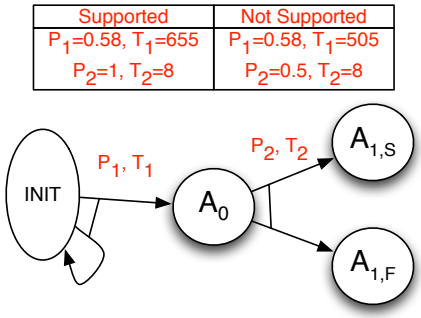


Figure 5.12: Insulin Pump: Device Injection Intervention

The injection of a malicious device can be supported by the destruction of the old one. This is modelled by assigning two different values to the transitions going from INIT to A_0 and from A_0 to $A_{1,S}$ (cf. Figure 5.12):

Supported: when the intervention is supported by *Destruction* the probabilities and the time rewards values are set accordingly to the results in Figure 5.5. Specifically: $P(INIT \rightarrow A_0) = 0.58$, $T = 655\,TU \approxeq 44\,min$ for INIT$\rightarrow A_0$, and $P(A_0 \rightarrow A_{1,S}) = 1$, $T = 8\,TU \approxeq 30\,sec$ for $A_0 \rightarrow A_{1,S}$.

Not Supported: when device injection it is not supported, $P(INIT \rightarrow A_0) = 1$, $T = 505\,TU \approxeq 33\,min$ for INIT$\rightarrow A_0$, and $P(A_0 \rightarrow A_{1,S}) = 0.5$, $T = 8\,TU \approxeq 30\,sec$ for $A_0 \rightarrow A_{1,S}$.

The probability and the time reward for $A_0 \rightarrow A_{1,S}$ is taken from the data regarding the probability extraction process at page 96.

Listing 5.5 shows the reward structure used to compute the time needed for the intervention while Figure 5.13 shows the probability to successfully inject a device as a function of time T (expressed in Time Units).

```
rewards "time"
    [supported] true: 655;
    [not_supported] true: 1;
    [injected] true:8;
endrewards

// Property for model checking

Rmax=? [F A_inj=2]
// A_inj=2 corresponds to the
// successful injection of a new device
```

Listing 5.5: Insulin Pump: Time Rewards for Device Injection

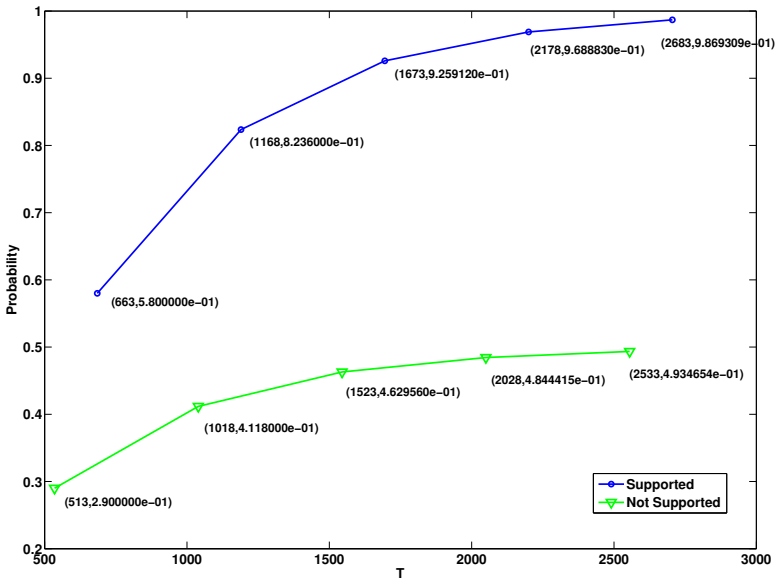


Figure 5.13: Insulin Pump: Success Probability for Device Injection

The results confirms that whenever injection is supported by destruction the overall probability of success is higher. Furthermore when the intervention is not supported, the success probability goes asymptotically to 0.5. As a minor remark, injecting a device in the supported case takes more time ($150\,TU \approxeq 10\,min$). This is in fact the time needed to destroy the device.

5.1.4 Discussion

This concludes the analysis of the insulin pump system with PRISM. There are some observations to be made regarding the analysis:

- The results throughout all interventions show the same behaviour. This is due to the fact that the insulin pump is a highly localised system (as BAN usually are), therefore localisation is a key action in all interventions, hence its influence is present in all data. Furthermore since the system is not protected with any other mean (e.g. encryption), localisation is the only "countermeasure[3]" that keeps the attacker away from attacking the system. If this were not the case the interventions that have to deal with encryption, such as *Data Knowledge*, would show a different behaviour globally.

- Interventions can be combined whenever they share a common action (e.g. device location) so that the attacker has to locate the device only once thus perform multiple interventions in less time.

- The model was analysed against a specific attacker: an attacker with a single eavesdropping device. This was done because analysing the system with a stronger attacker it is not that much interesting as it does not leave any margin to "counteract" the attacker. A stronger attacker would only lower the time to locate the device while increasing the probability. In this case the behaviour of the interventions (namely the curves in the figures) would be the same, only with higher probabilities and lower time.

As the reader has probably noticed, the model here has been used to explore specific cases for the system, the cases the analyst considered interesting and most important. From this analysis the application of the model to real scenarios should be clearer: it is not a kind of oracle that can be queried by the analyst to know whether a system is secure or not, quite the opposite it is a fine grained tool that helps in understanding what are the weaknesses of a system, given certain assumption made by the analyst himself, and more importantly where it is possible to modify the system in order to strengthen it.

[3]Here the term is clearly used improperly, as a passive action cannot mimic an active action such as the one that the concept of countermeasure recalls.

5.2 FleGSens: a WSN System for Area Monitoring

As specified earlier in Chapter 2, the FleGSens system [52] is a WSN for monitoring trespasses in an area under surveillance. It uses the iSense nodes, described in [19]. It is composed of two main kinds of devices: gateways and sensors. Gateways receive the transmissions from the sensors, organise data and relay them to the control centre. Sensors are equipped with an infrared sensor which detects movements at a range of 10 *meters* within a view angle of 110 ° (denoted "F" in Figure 5.14).

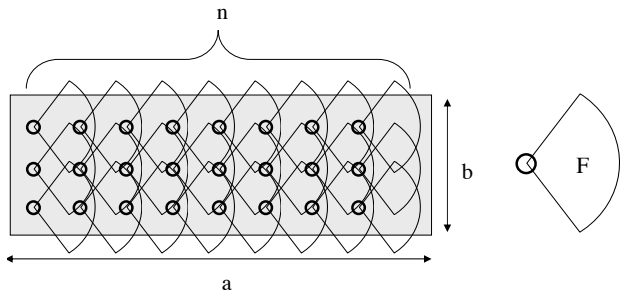


Figure 5.14: FleGSens Topology - Source [52]

Figure 5.14 shows the topology of the network: the sensors are deployed over a certain area in a strip fashion. The radio technology used complies with the IEEE 802.15.4 standard [3], working at a rate of $250Kbit/s$ with a range of 30 *meters*; the sensors make use of the AES block cipher as defined in the IEEE standard. Each sensor stores a preshared key with the gateways which is used to authenticate each message sent by the sensor. Cryptography is used only for authentication and not for Encryption, this means that data are sent in clear (as for the insulin pump) but they cannot be forged without knowledge of the key.

Within FleGSens there are two protocols in place that were designed to strengthen the system both in terms of efficiency and security: the "Trespass Detection Protocol" and the "Node Failure Detection Protocol". The first one organises the network in small clusters which gather collective information about trespasses and send it in one transmission only. This is done to prevent one event triggering traffic flooding, hence overloading the gateways and the overall network in general. Furthermore, to avoid false positives due for example to animals or similar sporadic events, a sensor recognises a movement as a trespass only if two movements are detected within a variable time usually set to 10 seconds.

Therefore if a sensor registers two movements more than 10 seconds apart, it disregards them. The second protocol is needed to detect node failures, so that failed nodes can be repaired or replaced. It is important to note that this protocol would mark as failed also nodes that have been destroyed or whose battery has been depleted or taken off.

In this analysis the analyst decided to focus on the sensors only as these are the ones that perform the movement detection. Furthermore the gateways are supposed to be deployed well within the network hence not reachable before raising an alarm.

5.2.1 Definition of Interventions

As we have seen most of the interventions in the case of the insulin pump, we just focus on three of them for this analysis:

- Destruction.
- Reprogramming.
- Energy Reduction.

Specifically they can be defined as follows:

Destruction: All states are active, the attacker can both destroy the device physically or remotely by making the sensor deplete its energy.

Reprogramming: As the sensors are implemented over a standard technology, the attacker may reprogram it. The paper does not specify any kind of remote reprogramming feature, hence the only way the attacker has to reprogram it is by physical access. The active states are those on the first branch.

Energy Reduction: FleGSens is a WSN, its devices have limited lifetime and energy, hence it is vulnerable to Energy Reduction. All the states are active as the energy could be reduced by taking out the battery or by forcing the sensor to deplete its energy by elaborating some data (cf. remote energy reduction, Chapter 4, Section 4.3). In this analysis the analyst explores only the second branch of the intervention (i.e. remote energy reduction) as the first branch (which models the device tampering) needs an analysis of the effectiveness of the infrared sensor to assess the successful tampering of the sensor without the attacker being detected. Furthermore if the attacker found a way to go undetected to the infrared sensor, it would not make sense for her to deplete the energy of the sensors

as she would already possess a method to render the detection network useless.

The other interventions were disregarded for the following reasons:

Eavesdropping: As the network is spread over a specific area (e.g. a national border) and it is meant to guard it, the attacker most likely knows its position. If we add the fact that the radio technology is a well known standard, and that the detection distance is considerable, we can fairly assume that the attacker is able to eavesdrop packets with a probability close to 1 every time that a packet is transmitted. Furthermore considering that this happens every 2 seconds (from the specifications of the network [52]) then it becomes even more clear that modelling eavesdropping in this case is not worth.

Data Knowledge: As data is transmitted in clear and given the considerations we just made about eavesdropping, it is not interesting to analyse this intervention for the purposes of this analysis. It is true that the attacker could get hold of cryptographic material with this intervention. But she should do this for every node in the network as the key is specific to each sensor. Furthermore technical specifications about the cryptographic mechanisms involved are missing: (key generation scheme, random numbers generators etc.), hence this intervention is not investigated.

Disturb/Partial Data Modification: as the cryptographic material is not known it is not possible for the attacker to forge a message. Furthermore reply attacks are mitigated by the use of authenticated time stamps, hence the attacker cannot use a legitimate message to Disturb the network.

Full Data Modification: It needs access to the device, the same considerations about *Energy Reduction* for the tampering branch hold.

Device Injection: A device has a preshared key with the gateways, hence the attacker would need to know one of this keys to inject a device in the network.

Energy Control: This does not apply as sensors do not have any energy harvesting capability.

In addition, the purpose of this work is to show how the proposed model can be used to analyse the security of a system, hence going through the same interventions as the ones already discussed for the insulin pump system would not have added anything significantly new.

5.2.2 Probability Extraction

In this section probability extraction for the actions relevant to the analysed interventions is discussed. These actions are:

- Device Location.
- Device Crashing.
- Memory Write Access.
- Device Physical Access.
- Remote Energy Reduction.

Physically Force Data Transmission is not discussed since the organisation in clusters inhibits the effectiveness of the action, furthermore the only way to stimulate a transmission is to trigger the infrared sensor. This would raise a trespassing alarm, and in this analysis we suppose that the attacker does not want to do that.

Device Location - As each sensor transmits a short packet every 2 seconds and the transmission range is considerable, the attacker (even a weak one) would locate a sensor within tens of seconds. Furthermore the attacker should know what the area under surveillance is and approximately where the sensors are placed. In addition sensors are placed 5 to 7 meters apart, finding one should not be difficult once the area is identified even at the naked eye. Hence the probability for device location can be set to one with a small time reward.

Device Crashing - Once a sensor has been found it should not be a problem to crash it. The only problem is to get hold of the sensor without being detected from the network. For this goal the clustering mechanism of the trespassing detection protocol could be exploited. Specifically the protocol is designed to filter out sporadic events, it does this by cancelling events which occur singularly within a short period (typically 10 seconds, cf. [52]). The attacker could exploit this by approaching the sensor with slow movements which occurs more than every 10 seconds. Unfortunately computing the probability for this requires the specific knowledge of the infrared sensor sensitivity to movements. Since device crashing was already explored in the analysis of the insulin pump, this action is discarded here.

Memory Write Access - This action is part of the reprogramming intervention. The iSense sensors can be programmed by direct programming interface, or, if equipped with a proper module called OTAP (Over-the-Air-Programming),

they can be programmed wirelessly. The FleGSens system is specifically built with the infrared sensor only (i.e. no OTAP) hence the sensors can be programmed only by direct interface. These devices do not provide any tamper proof mechanism or access restrictions to the memory. As a matter of fact the authors of [52] assert that it is possible for an attacker to compromise a sensor (i.e. reprogram it); they even suppose that the attacker could get hold of the cryptographic material in the node by doing this. This means that there is no mechanism to prevent access to the memory of the device, both data and program memory. Hence for the purposes of this analysis the attacker is supposed to have access to the memory with certainty (i.e. probability 1). The time needed depends on the attacker's knowledge on how the system works. If the attacker knows the system (and this is usually the case) then she should be able to reprogram the system in a matter of minutes.

Device Physical Access - This action is needed for *Reprogramming*, as to reprogram a sensor the attacker needs physical access. To get it, it has first to approach a sensor undetected, disable the infrared sensor and then physically access the device. As stated earlier the FleGSens system has a trespass detection protocol which detects a trespass only if two movements are detected within a 10 seconds window. Technical specifications which would help to understand what is considered a movement and what is not are not available, hence we will simply assume that with a movement it is possible to cover $50cm$ of space. If we consider that an infrared sensor can detect a movement $10m$ away, the attacker needs $10/0.5 = 20$ movements to reach the device and be able to disable the sensor (e.g. by covering it with an IR opaque material). If a movement takes 2 seconds the attacker needs $20 \times (2 + 10) = 400\,s$ which is roughly 7 minutes. The time reward for this action is then set to 7 minutes with a transition probability of 1.

Remote Energy Reduction - This is the most interesting action as it concern an intervention that it was not addressed before and that operates over the energy of the device. With this action the attacker stimulate the sensor activity thus making it reduce its energy. She can do this by transmitting data to the sensor, (any data) as a radio device in general consumes energy just by receiving transmissions. Specifically according to [56] the energy depleted per bit received in a IEEE 802.15.4 device at 250 $Kbit/s$ is 133.20 nJ/bit.

The attacker can choose to transmit three types of data:

I. A legitimate old transmission.
II. A forged transmission, which will be discarded as it is not authentic.

III. A malformed transmission, which does not satisfy the integrity check and is therefore discarded at the MAC layer.

With the first two types of transmissions the node loses energy by receiving and processing the packet as well. This processing consists in performing the cryptographic actions to validate the message. The energy cost for cryptography can be high if the cryptographic operations are performed by the processor. For common sensors, for example, this value ranges from 10 to 40 μJ for a single 128-bit AES block (see [24] for details). This is not the case for the iSense mote, since it performs all cryptographic operations on an hardware dedicated chip. In this case the energy consumption is in the order of tens of pJ/bit (these data are taken from [47]).

With the third transmission the node consumes energy simply by receiving the transmission. Even if data do not make sense for it, the sensor wastes energy by simply receiving them.

For this action the probability for the first two types of transmissions is set to 0.97 which is the probability of receiving a correct frame (it is the same as "fake data injection" in the case of the insulin pump). The probability is set to 1 in the case of the third transmission.

Considering that cryptography in the FleGSens system does not impact considerably on the energy budget of the sensor, we can fairly treat all types of transmissions as if they brought the same amount of energy depletion per bit received. If this were not the case, if for example a cryptographic operation would deplete energy in the order of μJ, an injected transmission which is not received correctly ($P = 1 - 0.97$) would still deplete the energy amount corresponding to the reception of the bit (as a transmission of the third type). As a minor remark, the system prevents replay attacks but does not check for them, hence replaying a transmission does not raise any alarm.

The iSense sensors are equipped with a CR2477 battery. This battery has a charge of 1000 mAh and a supply voltage of 3 V. Considering than $1Ah = 3600C$ this translates into:

$$1 \times 3600\,C \times 3\,V \; = 10800\,J$$

Hence the sensor has enough energy to receive:

$$\frac{10800}{133.2 \times 10^{-9}} \times \frac{1byte}{8bit} \times \frac{1}{2^{30}} = 9.44\,GByte$$

Given that the rate of transmission is 250 $Kbit/s$ the battery can be depleted in 84 hours. If you add the fact that other things may affect the battery life, then this time could even be smaller.

As in the case of eavesdropping the attacker should not find any difficulty in executing this attack as she has only to transmit arbitrary data. Furthermore she can boost the signal she transmits, thus increasing the range of the transmission and affecting more sensors at once.

This kind of attack goes undetected since the "Node Failure Detection Protocol" detects failed nodes when they have already failed (i.e. battery depleted).

Regarding the time unit, since FleGSens makes use of AES-128, and the minimum block size is 128 bit, the time unit is the time needed to transmit a AES block, hence $\frac{128bit}{250Kbit}\, s = 0.512\, ms$.

5.2.3 Analysis of the Model

The model can be analysed in a similar way to what has been done earlier for the insulin pump system. The results here will show fixed numbers and not probability of success as a function of time; this is due to the fact that there is not a temporal dependency in the interventions for the FleGSens system for two reasons:

- It is not a highly localised system and the range of devices is higher.
- Devices transmit at a high rate (every 2 seconds), hence the event of an eavesdropper passing by and not detecting a transmission is most unlikely.

Time comes in when the attacker depletes the energy of a device; we will see it in the analysis of the energy reduction intervention.

Energy Reduction - This is the first intervention to be analysed as its results are needed for *Destruction*; as a matter of fact Table 2.1 in Chapter 2 shows that *Energy Reduction* can imitate *Destruction* since when a battery is depleted the device is as good as destroyed. Specifically the results obtained in this intervention will be then used as input for *Destruction*.

Figure 5.15 shows the intervention with the inactive states. The transition probability $P(A_{1,S} \rightarrow A_{4,S})$ has different values depending on the type of transmission the attacker is forging. As already said above, in the case of FleGSens

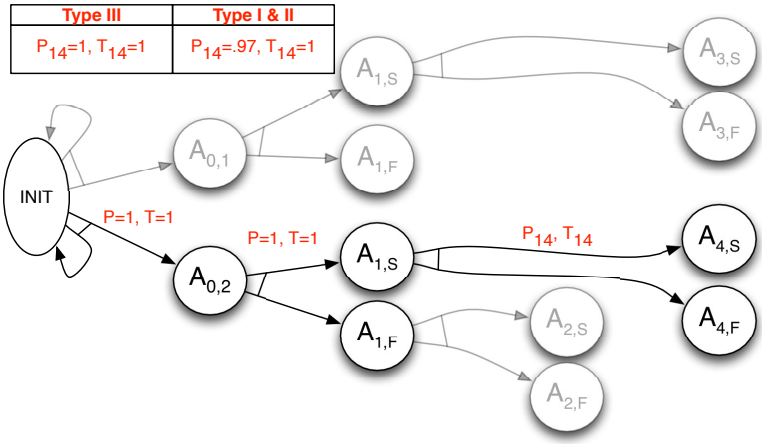


Figure 5.15: FleGSens: Energy Reduction Intervention

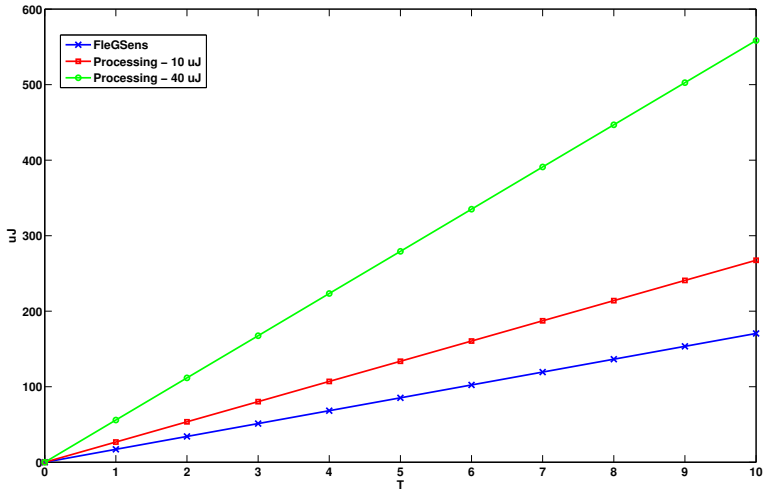


Figure 5.16: Energy Reduction of a Sensor with different values of Energy Depletion

it does not make sense to compute it as the energy depleted by processing the data is negligible with respect to the amount of energy depleted by receiving the bits. Anyway for completeness and for use in cases where the energy reduction

by processing is significant, data with different energy depletion values were computed. Figure 5.16 shows the value of the depleted energy as a function of time T (expressed in time units). As it is possible to see when the energy for processing increases, the amount of depleted energy increase as well, and noticeably. The values used for the processing energy are comparable with the ones in [24].

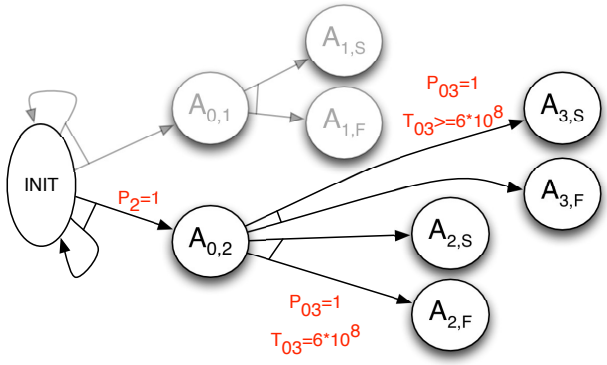


Figure 5.17: FleGSens: Destruction Intervention

Destruction - This intervention is modelled as in Figure 5.17: the active states are those that represent remote destruction of a device, this is achieved by depleting the energy of the sensor completely. As specified in the previous section the crashing of the sensor is not modelled as the attacker would be detected, hence invalidating the purpose for destroying the sensor itself.

In this intervention state $A_{2,S}$ represent the destruction of a single device, while state $A_{3,S}$ the destruction of multiple devices. They differ only in the time needed to accomplish the intervention. The time to deplete a sensor is 84 *hours* (see 116), which is the time reward for the transition $A_{0,2} \rightarrow A_{2,S}$ (84 *hours* $\approxeq$ $6 \times 10^8\, TU$[4]). When the attacker targets multiple nodes (state $A_{3,S}$), some of them could be on the edge of the transmission range, hence they would receive only a part of the transmissions aiming at reducing their energy, therefore the attacker has to take more time to deplete the energy of these devices otherwise they would have less energy but they would still be active.

Reprogramming - As already said in Section 5.2.2, *Reprogramming* can be performed only with a physical access to the device. Hence the active states are

[4]Remember that $1\,TU = 0.512\,ms$.

only the ones in the first branch.

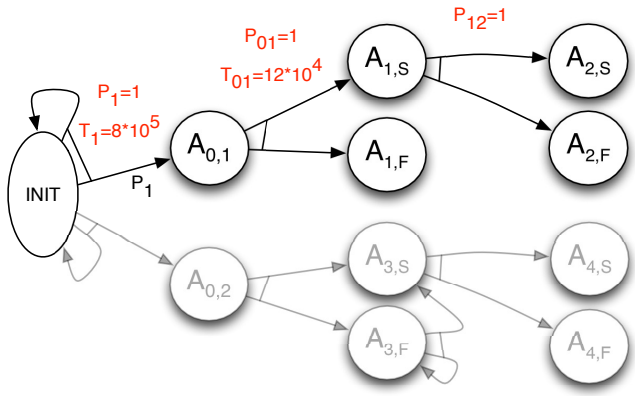


Figure 5.18: FleGSens: Reprogramming Intervention

Figure 5.18 shows the intervention with the related transition probabilities and time rewards expressed in time units. Specifically the transition $INIT \rightarrow A_{0,1}$ represents "Device Physical Access" (cf. page 114) and $A_{0,1} \rightarrow A_{1,S}$ "Memory Write Access" (page 113).

To successfully perform the attack the attacker needs around 8 minutes. With respect to this intervention and the FleGSens system there is an important observation to make: the analysis performs is valid for any attacker the analyst might consider, since there is no remote way to reprogram the node and the attacker wants to stay undetected.

5.2.4 Discussion

The way the analysis was performed in the case of the FleGSens system is quite different from the Insulin Pump System: in the latter the analysis showed a probability of success that is strongly dependent on the time the attacker has to perform the attack. For the FleGSens system this is not true anymore, at least for the interventions the analysis focused on. Here carrying out an intervention is a matter of time and not of probabilities. This is because device location is not a critical action anymore. This can be explained by observing that:

- The FleGSens system is a WSN surveillance system, spread throughout a

specific area, which is supposed to be known by the attacker (legitimate assumption as every surveilled area usually is identified by some sign).

- Devices transmit with high frequency (every 2 seconds), thus making device location trivial.

Usually WSN are designed so that they are energy efficient with minimal communication activity. Hence in a typical WSN locating a device would still pose similar problems to the insulin pump. FleGSens was designed to be energy efficient to some extent (the clustering organisation for example), but not on the communication level, this has consequences on device location and eavesdropping in general.

About the reprogramming of devices, the fact that there is no tamper mechanism that prevent access to the devices and the program memory seems not protected in any way, makes the reprogramming only a matter of time. The designer probably thought that an attacker would be detected by the sensor way before she had accessed it and perhaps did not perceive this as a serious threat.

There is another aspect that has not been discussed but that deserve to be mentioned: the system establishes routing paths that goes from each sensor to the gateways. Messages are forwarded along these paths. If the attacker managed to reprogram a sensor, she could tamper with the routing algorithm and create malformed path so that messages do not arrive at destination, thus bringing the system down all at once. The routing algorithm is not specified in the literature [52], hence a specific analysis was not possible, but still the threat exist and it is very serious in the opinion of the writer.

5.3 Summary

This chapter presents the quantitative analysis of the two case studies presented in Chapter 2. This is done in three steps: the construction of interventions, the probability extraction and the analysis using the PRISM model. The results of the first system show a similar behaviour across all interventions, this is due to the fact that these are based on the location of the target device, which is critical for the insulin pump system. For the FleGSens system the most interesting result is the destruction of a node by depleting its energy, which is the imitation of the *Destruction* intervention made by *Energy Reduction* (cf. Table 2.1).

Chapter 6

Final Remarks

This chapter summarises the results of the analyses performed in Chapter 5. Furthermore it lays out an iterative approach that the analyst can use to address the security weaknesses found in a system and it illustrates how the model can assist in the early steps of the secure design of ubiquitous systems. The last part of the chapter compare the work done in this thesis with the related work presented in Chapter 1.

6.1 Commenting the Results

In the previous chapter, two ubiquitous systems, namely the insulin pump system and the FleGSens system have been analysed. The analysis focused on specific interventions and actions that the author considered most interesting. Throughout the two systems analyses the interventions which were analysed in detail were:

- Destruction.
- Eavesdropping.
- Data Knowledge.

- Disturb/Partial Data Modification.
- Full Data Modification.
- Reprogramming.
- Node Injection.
- Energy Reduction.

The *Energy Control* intervention was not analysed as none of the two systems had energy harvesting capabilities, which enable a device to recharge its batteries. Nevertheless *Energy Control* and *Energy Reduction* are very similar, and the probability extraction process works in a similar way as well. The only difference is that in *Energy Control* the analyst has to accommodate for the recharge rate of the batteries, its timing and all the features that depends on the type of energy source. This should be slightly more difficult compared to the process performed for *Energy Reduction* (cf. page 115) but still on the same level of difficulty.

Comparing the results of the two analyses, the attacker's abilities are affected by three main aspects:

1. The dimension of the system, which affects device location.
2. The protection of transmissions with security properties such as confidentiality or integrity, which inhibit for example fake data transmission or eavesdropping.
3. The physical security of a device with mechanisms such as tamper proof.

With respect to the results given in Chapter 5, the first one produces the most noticeable effect. In the case of highly localised systems, like the insulin pump, the identification of the target and its localisation is the most critical action, the one that takes more time. On the other hand, with widespread systems, localising a device may not be a difficult task at all, it depends mostly whether the attacker is looking for a specific device, nevertheless it should not pose much of a problem. The first line of defence for an ubiquitous system therefore regards methods aiming to inhibit the localisation of devices, if not all of them, the most critical ones.

The protection of transmissions is another important measure to limit the attacker in her actions. The two analysed systems address this problem in two

different ways. The insulin pump system does not have any protection at all, just a CRC code to verify the integrity of a transmission and a PIN code, which is transmitted in clear and hence can be read by the attacker. There is no cryptographic key or mechanism which prevents the attacker to forge a message. This is a serious shortcoming and it should have been addressed in the design phase. The FleGSens system implements authentication, integrity and freshness mechanisms which ensure the origin of a message, its validity and that it was not modified in any way. The messages however are not confidential, hence an attacker may access confidential information, like where and when a trespass occurs, or even worst exploit the location protocol of the network thus building a map of the whole network. Considering the fact that the additional cost of implementing confidentiality is not high (the sensors already possess a key shared with gateways and the energy for encryption is negligible since it is done by a dedicated low power chip), the designers probably did not perceive this as a problem as it actually is.

With respect to the third aspect, the physical security of a device, both systems can be improved; the FleGSens system would benefit more from this. In fact in the insulin pump system all devices are on the body of the patient and physically accessing them may prove difficult. On the other side the FleGSens devices are deployed over uncontrolled areas, and they could be accessed easily, given that the attacker is not detected. Furthermore the FleGSens sensors provide a JTAG interface that ease the access to the programming and data memory of the device. Securing a device physically is not an easy task, anyway countermeasures should be put in place so that if a device is physically compromised, at least its rendered useless (by wiping clean its memory for example).

6.2 Addressing Security Through an Iterative Approach

As explained in Chapter 5 each analysis was performed in three steps: interventions construction, probability extraction and analysis. But after having analysed a system and identified the weakest parts of it, the analyst may decide to implement security patches and see how they affect the overall security. Figure 6.1 shows the flow diagram for this process. The original steps described in Chapter 5 are labelled as “Basic Steps”. For a bare analysis of the security of a system the analyst may just perform these steps. On the other hand, when a solution to possible security threats has to be produced, the analyst can rely on the process shown in the figure:

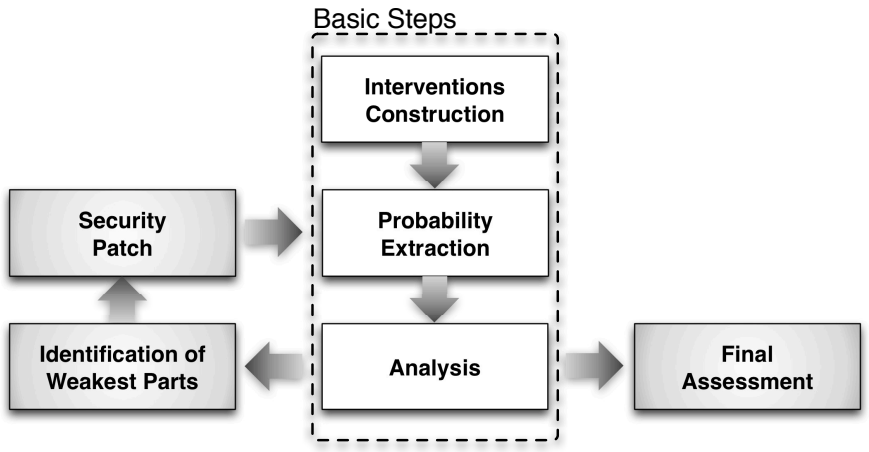


Figure 6.1: Analysis and Patch Process Flow Diagram

1. Identify the parts of the system which are weak.

2. Devise a security patch.

3. Perform the whole analysis from the probability extraction again.

4. Produce a final assessment of the security of the system, when he is satisfied with the analysis or when he deems it right.

What happens in this process is that the probabilities change and the weakest point moves from a part of the system to another. Furthermore by performing this iterative process, the analyst may uncover new vulnerabilities which were ignored in the first analysis since the weaknesses of the unpatched system posed a higher risk and hid the newly discovered vulnerabilities.

As an example we can consider *Device Injection* in the insulin pump case, looking at the results in Figure 5.13 injecting a device in the "not-supported" case can be neglected with respect to the "supported" one. When the latter is addressed by a security patch then the former may become of greater importance and the analyst may decide to address it as well.

6.3 Using the Model in the Design Phase

Having a method to analyse the security of an ubiquitous system efficiently and effectively is indeed very important. Nevertheless the case studies proposed outline how the lack of implementing security from the very first moment, the design phase, has made the system vulnerable, especially the insulin pump system. If the designers were made aware of the attacker model, even to a qualitative level, they would have probably produced a more secure and robust system.

The writer strongly believes that one of the main objectives that a security researcher should have, is to make people aware of security: what security is, what are its advantages, what are the different features that compose it and what are the risks that incur when it is neglected. This work, and specifically Chapter 2, apart from people belonging to the security community, is also addressed to people who wish to learn about security and specifically attacker capabilities.

As a matter of fact, the proposed model can be used in the design phase of an ubiquitous system to identify what are the parts of the system that needs to be secured and also what are the ones that do not pose any significant threat and that may be left as they are.

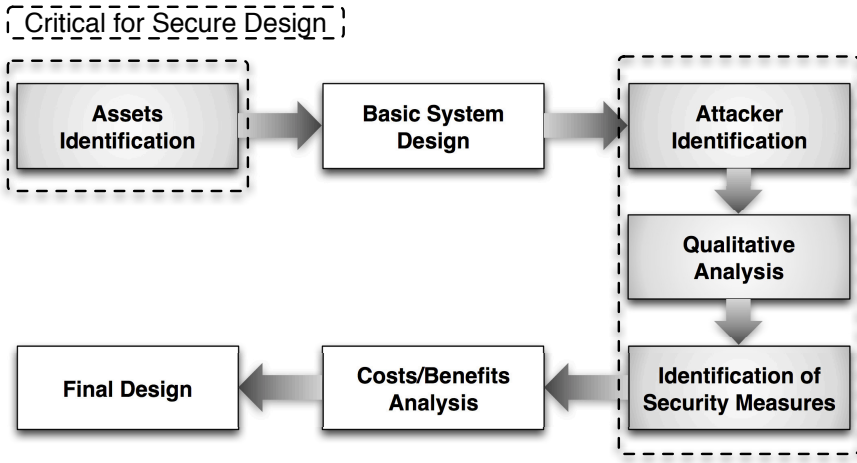


Figure 6.2: Securing a System by Design

There is no specific way to do this, any system designer can find its own; a possible process is illustrated in Figure 6.2. It consists of the following steps:

1. Identification of assets that need to be secured.
2. Design of a system with no security.
3. Identification of the Attacker: Strength, Presence and Time.
4. Qualitative evaluation of the system against each intervention and the identified attacker.
5. Definition of proper security measures.
6. Analysis of the costs and the benefits of implementing them.

With the first step the designer identifies what are the assets that need to be secured. The word asset here identifies both physical or abstract entities like for example:

- A device (e.g. a glucose sensor).
- Data which have to be confidential or whose integrity must be preserved.
- Functionalities of the system whose outcome needs to be ensured so there is no way for a malicious party to tamper with them.
- Other assets specific to the system's purpose.

It is not always true that all assets needs to be protected, if for example data confidentiality is considered of greatest importance while the communication system is not, then a designer may want to implement only traffic encryption, neglecting the physical security of the communication infrastructure. If then availability and real time communication becomes a strong requirement then the designer may also consider the physical security of the devices.

With the second step the system designer specifies a basic system with no security which implements all the functionalities and the entities required by the target system.

With the third step the designers begin to explore the security dimension of its target system. He does this by defining the type of attacker that may be interested in attacking the system, namely what kind of resources she may have and her strength with respect to time and presence.

In the fourth step the qualitative model defined in Chapter 3 comes into play. Specifically the designer has to perform a qualitative analysis of the interventions against the attacker defined in the previous step and the system itself. This analysis is very similar to the one performed in Chapter 3, Section 2.3. During this step the designer should always keep in mind the three aspects defined in Section 6.1, namely the dimension of the system, the protection of transmissions

and the physical protection of devices. This analysis has to be performed against every device of the system.

In the fifth step, given the weaknesses found with the analysis, the designer has to define appropriate changes to the design which implement the correct security measures. This process could easily end in defining new assets or new entities which carry out only tasks related to the security of the system. These can be for example authentication servers, secure tokens, Intrusion Detection and Prevention Systems, Firewalls etc. In addition such security measures may also regard other aspects which are collateral to the design, such as monitored physical access to devices (e.g. devices stored in secure locations which can be accessed only by authorised personnel), tamper proof mechanisms, or other types of physical measures. It could also be that a weakness of the system can be addressed simply by choosing a different technology, for example the use of random spread spectrum frequency hopping mechanisms to counteract eavesdropping.

Finally, after the security measures have been identified, the designer may have to perform an analysis of the costs and the benefits, both from an implementation and an economical point of view, and decide for the final design of a system accordingly.

The steps highlighted in Figure 6.2 are considered critical with respect to a secure design of the system for the following reasons:

- The correct identification of the assets to protect is clearly a crucial step, as neglecting to recognise key assets may endanger the whole system.
- The identification of the attacker who may attack the system is also of major importance, as the security measures put in place maybe insufficient if the identified attacker were weaker than the real one.
- The qualitative analysis has to be performed thoroughly otherwise vulnerabilities may be incorrectly addressed or be left not addressed.
- By identifying security measures the designer specifies how the threats are addressed, hence these need to be correct for he system to be secure.

6.4 Considerations About Related Work

In Chapter 1 Section 1.4, the relevant related work has been introduced and discussed. In this section the writer is going to connect the work done in this

thesis with the related work by discussing similarities and new aspects with respect to each of the work cited in Section 1.4.

With respect to the Dolev-Yao model [29], the attacker defined in this thesis has all the abilities the Dolev-Yao has, although she is limited in her actions: The Dolev-Yao was an omnipotent and omnipresent attacker, here the attacker is limited by its strength in presence and time, she cannot perform multiple actions together, very much like the model defined by Creese et al. in [23] and [22], and some of the actions may even be impossible. This limitations are accounted for during the probability extraction; as a matter of facts this process deals with the actual abilities of the attacker, taking into account her shortcomings. As an example take an attacker in a wireless network who wants to inject data and who has only one wireless card; in this case the attacker cannot eavesdrop and inject data at the same time, and this is indeed taken into account by the analyst. Furthermore the structure of the PRISM model itself allows for a transition at a time; multiple concurring actions have to be explicitly modelled, hence the analyst has to decide whether or not the attacker may perform two or more actions at the same time.

Considering the Bellare-Rogaway [13] and the Rubin-Shoup model [55], which provide the attacker with the ability of recovering session keys, the attacker model defined in this work has a similar approach as it provides the attacker with the ability of getting hold of cryptographic material as well. These two models are indeed focused on checking protocol security, nevertheless, with respect to the Dolev-Yao model, the corresponding real world these two models refer to is closer to the one this thesis refers to as well. Furthermore the idea that cryptographic material can be extracted from the memory of devices is very similar to the probabilistic oracles defined in the Rubin-Shoup model (cf. Section 1.4).

The work done by Basin et al. in [10] and [11] connects to this thesis as it checks the correctness of "physical security protocols". This is done by considering features like time and localisation, which limits the attacker in her ability to compromise the system. This thesis extends this features in a more broad and comprehensive way by defining attacker's presence and time with respect to her strength. In addition this extension bridge over the physical world and enables the attacker to actions such as gaining physical access to a device. This connects [10, 11] to [12], all three by Basin et al., where the authors consider the possibility of compromised devices resulting in leakage cryptographic keys at a protocol level.

As a final consideration the work presented in this thesis can be put in rela-

tionship with the science of *Risk Analysis* [1], which concerns with evaluating the probability of some dangerous events to happen. This thesis resembles risk analysis very much, specifically in the process of probability extraction and the quantitative analysis of the model.

With respect to the above mentioned related work, this thesis has specific major contributions:

- The identification of the attacker's actions, considering her spatial and temporal boundaries with respect to her strength.
- The definition of a quantitative model and a methodology to analyse the security of ubiquitous systems.
- The definition of the probability extraction process, which provides the quantities for checking the model.
- The implementation of a model to compute eavesdropping probabilities in a scenario where eavesdroppers and targets are mobile.

To the writer's knowledge this is the first work that aims at producing a quantitative attacker model in such a comprehensive way as it has been done in this thesis.

6.5 Summary

In this chapter the writer discussed the results of the analyses presented in the previous chapter. With respect to such analyses, the attacker abilities are mainly affected by three aspects: the system dimension, the protection of transmissions and the physical security of the devices composing the system. Afterwards an iterative approach to address the security threats of a system by using the model was illustrated. This process iterates the analysis with the application of security patches to the system. In addition, motivated by the fact that the analysed system presented severe security flaws, a methodology to help system designers in implementing security in the early phases of system design was presented. At the end, in the final part of the chapter, the writer makes a comparison of the work presented in this thesis and the related work described in Chapter 1.

[1] If the reader whishes to know more about the subject please refer to [53].

Chapter 7

Conclusions

Technology has experienced, in the last ten to twenty years, an incredible development where the act of computing has gradually moved from desk size devices to small computing devices which "weave themselves in the fabric of everyday life" (*Mark Weiser*, [63]). Smart buildings, automotive, devices for health applications, environment monitoring, all these technologies make use of an ubiquitous computing component.

In this thesis the problem of defining an attacker model for Ubiquitous Systems has been addressed and a method to apply it to real cases has been developed. The attacker is defined by three dimensions: a) *Interventions*, b) *Presence* and c) *Time*. These define: a) what the attacker can do, b) what space she can influence and c) for how long she influences the system. We call this attacker the "Cyber Physical Attacker" as she can attack a system from a physical perspective as well as a cyber one. The attacker is also defined by her strength which is directly connected to presence and time. Up to this point the model describes the attacker from a qualitative point of view.

To use the model for a more thorough analysis, the author developed an implementation of the model in PRISM [42], a probabilistic model checker for systems that express both probabilistic and non-deterministic behaviour.

In order to perform a quantitative analysis of the system, the analyst has to

perform three steps:

1. Construction of interventions.
2. Probability extraction.
3. Analysis of the PRISM model.

The first specifies how the interventions apply to the system under analysis, probability extraction computes the probabilities that are used as input for the PRISM model, and the final step is the analysis itself using PRISM.

The most critical step is the one of *Probability Extraction*. It is also the most time consuming one. In this process the analyst has to find the probabilities for the single actions that compose interventions. In doing this, he could make use of other models, such as the mobility model the author developed (cf. Chapter 4). This model computes the eavesdropping probabilities in a scenario where attacker and target move within a defined space, according to a pattern which can be both probabilistic or deterministic. The analyst could use also models which predict the status of the energy reservoir of a device either by modelling its batteries or its energy resources in general. The analyst's experience plays an important role as well. As ubiquitous technologies differ very much from each other, the approach used for extracting a probability for a specific action could be completely different from system to system. The two case studies proposed in Chapter 5 for example have two different ways to address "Device Physical Access". In the FleGSens system the analyst makes use of the infrared sensor specifications to compute the time the attacker needs to access the device whereas, in the insulin pump, it is more a social problem since it is a matter of stealing a device from someone. If you think about the fact that these are two very different technologies, the first is a WSN for area monitoring while the second is a WBAN for health applications, then the fact that the analyst has to use two different approaches does not come as as surprise since one system is meant to interact directly with people whereas the other carries out an unsupervised task.

From the analysis performed in Chapter 5, we have noticed in Chapter 6 that the attacker abilities of attacking an ubiquitous system depend greatly on three aspects:

1. The physical dimension of the system.
2. The protection of transmissions.

3. The physical security of the devices.

These aspects emerged prominently in the analysis of the two proposed systems but, given the manifold nature of ubiquitous systems, other different aspects may be influential for other cases for example the nature of the energy source in an energy harvesting system.

In Chapter 6 in addition we have seen that the model can serve two purposes: first it can be used as a tool to help the analyst to discover what and where are the weaknesses of a system and possibly how to address them, and second as a support tool for system designers who wish to implement security form the very beginning of their design. One makes use of the quantitative model while the other uses the qualitative one. Table 7.1 summarises these two purposes.

	Qualitative Model	**Quantitative Model**
Chapters:	2	3 and 4
User:	System Designer	Security Analyst
Application:	Used during the design phase to identify where and to which extent implement security.	Used as a support tool for the analysis of the security of a system.

Table 7.1: Purpose of the Model

With the quantitative model, the analyst decides which parts of the system he wants to analyse and then he constructs the interventions and performs the analysis accordingly. The analyst has to choose what kind of attackers he wants to perform the analysis against, specifically what is their strength in terms of time and presence; this will then have influence on the probability extraction process. As we have seen in Chapter 5, the usage of the model can be very fine-grained depending on the purposes of the analyst. Furthermore the analyst can use an iterative approach where at each iteration he corrects the security weaknesses found and look for new ones that may arise. This process terminates when the analyst is satisfied with the overall security of the analysed system.

The qualitative model can be used to help system designers and engineers to implement security from the beginning in their system. In Chapter 6 the author explained how the model description given in Chapter 2 could be invaluable for people who have little or no knowledge about securing a system and who are called to design systems where security can be critical (the example of the insulin pump is striking). Unfortunately it is not rare that people that are called to design IT systems have little knowledge about security or, even worse, that they implement it in a very wrong way. This happens often, especially in the sector

of health care, environment monitoring and in the automotive industry. In these sectors in fact, the focus is on the main requirements: safety of the equipment and the user, real-time behaviour, energy efficiency, operational availability etc. hence the security of the system itself is somehow set aside; without thinking that one attack may destroy the system, invalidating all the main requirements at once. Part of the problem, as [21] stresses in the case of medical applications, lies in the fact that the computer security community is often disjoint from the other communities, especially in the biomedical sector and to some extent within the wireless sensor network community as well. Part of the problem is also that designers are usually not aware of security in general, hence they rely on technologies which are known to be vulnerable to a series of threats (such as for example Bluetooth [30]) and whose implementation in security critical applications should be carefully considered.

A summary of the contents and the main contributions for each chapter follows:

- Chapter 2:
  1. Definition of the attacker model.
  2. Definition of the relationship between dimensions.
  3. Qualitative analysis of two case studies.
- Chapter 3:
  1. Quantitative implementation of the model.
  2. Presence and Time in the model.
- Chapter 4:
  1. Definition of Probability Extraction.
  2. Definition and Implementation of a mobility model for eavesdropping.
  3. Practical example: eavesdropping the insulin pump.
  4. Guidelines to probability extraction.
  5. Discussion over the analyst role in probability extraction.
- Chapter 5:
  1. Qualitative analysis of two case studies.
- Chapter 6:
  1. Discussion over the results of the analyses performed in the previous chapter.
  2. Description of a method to address the security of a system through an iterative approach.
  3. Discussion on how the model can be used in the design phase.

each

Future Work

There are three main topics for future work:

1. Augmentation of the model with a study on the willingness of the attacker to attack.
2. Investigation of a way to make the process of probability extraction more automatic.
3. Study of how the use of different communication technologies impacts on the attacker capabilities.

1. During the analysis of the model, we have encountered situations where the attacker had two or more possible ways to achieve an intervention, each carrying its own benefits as well as risks. In this work we have been interested only in computing the success probability of every action the attacker might perform. To take a step further, the willingness of the attacker to carry out these actions should be investigated. There are already several studies that focus on this aspect, and most of them are based on game theory (for a reference on network security and game theory see [4]). The basic idea is to model a game with two players: the attacker and the defender. Each of them has a strategy unknown to the other and they can make the following decisions:

- The attacker decides whether or not perform an attack, based on the knowledge of what are the benefits of the attack (i.e. what she gains) and what are the risks (i.e. what happens if she is caught).
- The defender chooses how strongly he wants to defend the system, based on the cost of the defence (i.e. what are the losses if the attacker wins) and how much effort he has to put into defending the system.

Given this analysis, the model may be augmented in such a way that the most likely actions the attacker might perform are singled out and highlighted to the analyst who can then focus on them. Furthermore actions which are very unlikely could be discarded at will, thus simplifying the analysis.

In [38] a connection between game theory and attack-defence trees is defined (for attack defence trees see Chapter 1, Section 1.4). These are powerful graph representations of security threats that help in finding attack-defence patterns.

The use of attack-defence trees in connection with game theory should be investigated as it could improve and simplify the definition of the games, bringing to more significant results.

The use of game theory would help the analyst to understand what are the parts of the system worth analysing or worth defending, and what are the actions the attacker most likely would take.

2. Probability extraction is a key process as it decides the probabilities to input into the PRISM model. We have seen in Chapter 4 that this process can be generalised for some actions, particularly the author explained the way to extract probabilities for eavesdropping. Other actions unfortunately are strongly dependent on the technology the system under analysis is composed of, hence the analyst has to rely on other methods and mainly on his experience to perform probability extraction (cf. Table 4.4). In order to provide the analyst for a more effective analysis for such actions, probability extraction should be further investigated, a classification of the technologies available for Ubiquitous Systems would be a logical step towards a study on extracting probability from them. Furthermore, as the usage of energy harvesting technologies is going to be more and more common, a study of the security implications with respect to those actions that aim at exploiting the energy of a device, should be carried out.

3. The development of new technologies has addressed several threats that were discovered in the past, but it also opened new ones. Considering EH-WSNs, the example made in Section 2.2.2, at page 24, is a striking one; the fact that a EH-WSN node has an amount of energy that can increase could raise new threats that were not present in the earlier technology, when not dealt with accordingly. As another example, considering the problem of eavesdropping in the case of the insulin pump system, the use of body coupled communication (BCC, see [8]) would have mitigated the threat posed by an eavesdropping attacker.

Given these examples a study of the security implications that the application of different wireless technologies have, would add details to the attacker possibilities of performing specific actions, more importantly it would give system designers an important tool to evaluate the security requirements of their system against underlying technologies. A good starting point would be to start from the standard technologies presented in Chapter 1 (see Figure 1.2), and draw a list of the security properties that each technology offers, and their applicability to ubiquitous systems.

Appendix A

Prism Language

This appendix contains a resumé of the PRISM language. For a thorough reference please visit `www.prismmodelchecker.org`.

A.1 The Modelling Language

The PRISM modelling language can model four types of Markov models already defined in Chapter 3:

1. Discrete Time Markov Chains.
2. Continuous Time Markov Chains.
3. Markov Decision Processes.
4. Probabilistic Timed Automata.

The type of model is usually specified at the beginning of the model description with one of the following keywords:

```
dtmc || ctmc || mdp || pta
```

A model can then have global variables. These can be either integer or Boolean. They are defined with a statement like:

```
global y : [1..10] init 1; //integer
global b : bool init true; //boolean
```

Constants can also be defined. Beside integer and boolean these can also be double (i.e. decimal) values. They are defined by a statement like:

```
//constants
const int radius = 12; //integer
const double pi = 3.141592; //double
const double area = pi * radius * radius;
    //double value computed by the given
formula const bool yes = true; //boolean value
```

By the example it is possible to see that constants can be defined using other ones.

The core of a PRISM model is called “module”. It is defined by a set of local variables which represent its local state and a set of commands which express its behaviour. Each command has a label, a guard and an update part. The label is used to synchronise actions across different modules, a guard is a predicate that needs to be satisfied in order to execute the command and the update describe the new values the variables assume and with which probability or rate. A module is defined by a statement like this:

```
module dummy
        x:[0..6] init 0; //local variables

//Commands
[]       x!=6 -> 1:(x'=x+1);
[reset] x=6  -> 1:(x'=0);
endmodule

//Command structure:
[label] guard -> rate:(update)
[label] guard -> probability:(update)
```

A PRISM model can be augmented with formulas. These are used to simplify the model definition within the modules. A formula can be composed of a mathematical expression, a boolean expression or some basic functions such as log, min or max[1]. A formula can be defined by the following statements:

```
formula double_x = x*x;
formula Max = max((x1-x_s1>0 ?x_1-x_s1:x_s1-x_1),
(y_1-y_s1>0?y_1-y_s1:y_s1-y_1));
```

A.2 Rewards and Properties

A PRISM model can also be augmented with Rewards structures. These assign reward values for a transition or for the time spent in a state. They are defined by a statement like:

```
rewards "NameOfReward"
    true : 1;
    x=3 : k*2;
    [reset] true : 1000;
endrewards
```

This reward structure behaves differently when the time is discrete or continuous. In the first case it assigns:

1. Line 1: it assigns 1 for every state visited.
2. Line 2: it assigns k times 2 for every state where x equals 3. k can be both a variable or a constant.
3. Line 3: it assigns 1000 for every transition labelled "reset".

When the time is continuous:

1. Line 1: it assigns 1 times the time spent in the state for each state visited.
2. Line 2: it assigns k times 2 times the time spent in a state for each state where x equals 3. k can be both a variable or a constant.

[1]For a complete list refer to `http://www.prismmodelchecker.org/manual/ThePRISMLanguage/Expressions`.

3. Line 3: has the same behaviour as the previous case.

A PRISM model can than be checked against three type of properties identified by the operator they use:

1. The P operator.
2. The R operator.
3. The S operator.

These have been already explained in Chapter 3.

The P operator computes a probability of an event occurring. The R operator computes properties based on reward structures. The S operator computes the steady state probabilities of the model.

These are specified by:

```
P query [ pathprop ]
S query [ pathprop ]
R query [ rewardprop ]
```

The query of operators can either express a bound on a value or ask for a quantitative result, e.g.:

```
P<.8   [ F x=1 ]
P=?    [ F x=1 ]
```

The first computes whether or not the probability of x being equal 1 is less than 0.8, while the second asks for the exact probability.

The “pathprop” assume different values for the P and S operator. In the latter it simply refer to a state of the model, in the first it can describe different properties, these are:

- X : ”next”
- U : ”until”
- F : ”eventually” (sometimes called ”future”)

- G : "always" (sometimes called "globally")
- W : "weak until"
- R : "release"

For a complete description please refer to `http://www.prismmodelchecker.org/manual/PropertySpecification/ThePOperator` .

The R operator works differently from the other two, as it expresses reward based properties. Specifically "rewardprop" can assume 4 different values:

1. reachability reward": F prop
2. "cumulative reward" : C¡=t
3. "instantaneous reward" : I=t
4. "steady-state reward" : S.

The first one is used to compute the expected reward to reach a specific state. The second one is used to compute the reward accumulated within a time t. The third reports the value of the reward at the instant t. Finally the fourth one gives the value of the reward at the steady state.

Appendix B

Mobility Model

This appendix contains details about the model developed to compute eavesdropping probabilities in a scenario where both the eavesdropper and the target station are moving.

The model is built and analysed with the PRISM tool. The PRISM model can easily reach a thousand lines of description, hence a program to generate the PRISM model file has been implemented using MATLAB [45]. MATLAB is a framework for numerical computation and visualisation which makes use of extended libraries, among which are libraries for mathematics and statistics. It has a programming language which resembles C/C++ in its structure although it is substantially different: it is interpreted by the MATLAB framework which deals with memory management, type assignments and similar so that the programmer can focus only on the algorithm, leaving typical problems, such as segmentation faults or exception handling, to the framework.

MATLAB is the choice that best suites the purpose, since it provides strong and complex mathematical functions which ease the implementation of the different probabilistic features the model has.

B.1 The Model Generator

The model generator is implemented as a MATLAB function which takes the following parameters as input:

1. X-by-Y space dimension.
2. The file name of the prism model.
3. An X-by-Y matrix defining the area the target can move over, called `target freedom`.
4. The initial position of the target and if needed also a most likely position.
5. An X-by-Y matrix defining the area the eavesdropper can move over, called `eave freedom`.
6. The initial position of the eavesdropper.
7. A boolean value which tells whether the eavesdropper follows a sequential path or a random one.
8. Transmission and retransmission probabilities: $P(Tx|NoTx), P(Tx|Tx)$.
9. Eavesdropping probabilities by distance between the eavesdropper and the target.
10. The step size for both the eavesdropper and the target.

The matrix of the areas covered by the eavesdropper has two forms: if the movement is sequential the path is identified by a series of sequential numbers ranging from 1 to a number representing the length of the path. If the movement is random, zeros represent points where the eavesdropper cannot move over, while ones points he can. The target mobility is purely random hence his matrix can be only in the form of zeros and ones.

The model generator supports an arbitrary number of eavesdroppers and targets, nevertheless increasing them easily produces models that cannot be analysed with PRISM, this is due to their size in terms of number of states and transitions.

B.1.1 Target Movement

In this model the target can be set to move in two ways:

1. Purely random: the movement probability is sampled from a uniform distribution.
2. Normally distributed over a space, with a focal point.

With the first, the movements of the target are modelled using a probabilistic distribution which is uniform in every direction. With the second one the target moves within an area, and his movement is biased towards a target point. This is modelled with a normal 2D distribution with the mean centred in the target point. This is exactly how the patient's movements were modelled in Chapter 4 (see Figure 4.3).

B.1.2 Eavesdropper Movement

The eavesdropper can move in two ways:

1. Sequentially, following a path.
2. Randomly following a probabilistic distribution.

The first refers to a scenario where the eavesdropper is searching an area following a specific path. This would obviously be the normal strategy for most the eavesdroppers. The second movement is similar to the one the target makes, the eavesdropper moves following a probabilistic distribution.

As MATLAB provides a complete probabilistic framework, the model can be easily augmented to include any kind of probability distribution may be required for the movement of the eavesdropper or the target.

B.1.3 Following the Target

The case were the eavesdropper follows the target (cf. Chapter 4, Section 4.2.3) is modelled in a different way. The space is smaller, defined by the eavesdropping

range, for example a 5x5 space, and even though both in the real world target and eavesdropper are moving, since the target is followed by the eavesdropper, its position can be modelled as fixed in the centre while the eavesdropper moves throughout the whole space in a probabilistic fashion. In this thesis we focused on the case when the eavesdropper moves following a 2D normal distribution centred on the target position, this to model the fact that he wants to be always as close as possible to the target. Indeed other probability distributions can be freely chosen depending on the case under analysis.

B.2 Generating the PRISM code

After whole the paramenters have been set or computed, the model generator builds a matrix that for each moving entity (i.e. eavesdropper or target) and for each point in the space, contains the transitions probabilities for every movement. Afterwards the actual file generation begins.

The model is a Discrete Time Markov Chain. Each entity is composed of two independent modules:

- Target:
  - Movement module.
  - Transmission module.
- Eavesdropper:
  - Movement module.
  - Eavesdropping module.

The firs one models the movement of each entity. The second one models the transmission and the eavesdropping. These two modules are syncronised in the actions so that everytime the target transmits, if the eavesdropper is in range, he eavesdrops a transmission according to the eavesdropping probability given for the distance between the two.

The distance is computed by a formula (cf. PRISM language Appendix A) which is generated by the model generator.

At the end the model generator provides also the reward structures needed to check the model.

Appendix C

Abstracts of Published Papers

Toward a Threat Model for Energy-Harvesting Wireless Sensor Networks

Security is a crucial matter for Wireless Sensor Networks. With the recent introduction of Energy-Harvesting nodes, it has gained even more importance. By exploiting the ability of scavenging energy from the surrounding environment, the lifespan of a node has drastically increased. This is one of the reasons why security needs a new take in this topic. Traditional solutions may not work in this new domain. Brand new challenges and threats may arise and new solutions have to be designed. In this paper we present a first taxonomy of attacks, focusing on how they change in the energy-harvesting context compared to regular sensor networks. We also discuss existing security solutions specific for the energy harvesting world and comment on the trend that this topic may follow in the future. Finally, we draw a comparison between the cyberphysical attacker we define in our model and adversary models belonging to security protocols verification literature.

Alessio Di Mauro, Davide Papini, Roberto Vigo and Nicola Dragoni
4th International Conference on Networked Technologies, Dubai 2012

Introducing the Cyber-Physical Attacker to Energy-Harvesting Wireless Sensor Networks

Cyber-Physical Systems based on Wireless Sensor Networks are pervading our everyday life, ranging from industrial to military applications. Due to the criticality of the tasks performed by these systems, and to the increasing number of fields in which they are employed, their security is a central concern. What is more, with the recent introduction of Energy-Harvesting nodes, securing such a system is even harder. An EH-node is able of scavenging energy from the surrounding environment, thus extending its lifespan, but this new feature introduces a new target for attackers. Traditional approaches to WSN security may not work in this new domain: new solutions have to be designed to cope with brand new challenges. A taxonomy of attacks is presented in this paper, which focuses on highlighting the novelties of the energy-harvesting context compared to regular sensor networks. We also discuss existing solutions specific to the energy harvesting world, and comment on the trend that this topic is likely to follow in the future. Finally, we sketch a comparison between the cyber-physical attacker we define and adversary models belonging to security protocols verification literature.

Alessio Di Mauro, Davide Papini, Roberto Vigo and Nicola Dragoni
Journal of Networking Technology, Volume 3, Number 3, September 2012

Security Challenges for Energy-Harvesting Wireless Sensor Networks

Security is a crucial matter for Wireless Sensor Networks. With the recent introduction of Energy-Harvesting nodes, it has gained even more importance. By exploiting the ability of scavenging energy from the surrounding environment, the lifespan of a node has drastically increased. This is one of the reason why security needs a completely new take in this topic. Traditional solutions may not work in this new field. Brand new challenges and threats may arise and new solutions have to be designed. In this paper we present a taxonomy of attacks focusing on how they change in the energy harvesting scenario compared to regular sensor networks. We also discuss existing security solutions specific for the energy harvesting world and comment on the trend that this topic may follow in the future.

Alessio Di Mauro, Davide Papini and Nicola Dragoni
2nd International Conference on
Pervasive and Embedded Computing and Communication Systems, Roma 2012

Lightweight MAC-Spoof Detection Exploiting Received Signal Power and Median Filtering

Abstract. IEEE 802.11 networks are subject to MAC-spoof attacks. An attacker can easily steal the identity of a legitimate station, even Access Points, thus enabling him to take full control over network basic mech- anisms or even access restricted resources. In this paper we propose a method to detect this kind of attack based on signal power monitoring. The main contribution of our work is the introduction of a median fil- ter that enables the detection of the attack by looking at the variance of the signal power. We take into account two types of references for the samples, time and number of frames, and compare their detection capabilities. Our experimental results show that the spoofing attack is successfully detected with both type of references. Moreover the median filter helps to reject false positives.

Davide Papini
5th Nordic Workshop on Dependability and Security NODES11, Kgs. Lyngby
Int. J. Critical Computer-Based Systems, Extended Version, To Appear

$(SC)^2$: a System to Secure Off-Card Contract-Policy Matching in Security-by-Contract for Open Multi-Application Smart Cards

The Security-by-Contract (SxC) framework has recently been proposed to support software evolution in open multi-application smart cards. The key idea lies in the notion of contract, a specification of the security behavior of an application that must be compliant with the security policy of the smart card hosting the application. In this demonstration we show $(SC)^2$ (Secure Communication over Smart Cards), a system developed to address a key issue of the SxC framework, namely the secure outsourcing of the SxC contract-policy matching service to a Trusted Third Party (TTP). $(SC)^2$ secures the communication between a smart card and the TTP that provides the SxC matching service.

Nicola Dragoni, Eduardo Lostal and Davide Papini
12th IEEE International Symposium on
Policies for Distributed Systems and Networks, Pisa 2011

$(SC)^2$: Secure Communication over Smart Cards How to Secure Off-Card Matching in Security-by-Contract for Open Multi-Application Smart Cards

TheSecurity-by-Contract(SxC) framework has recently been proposed to support software evolution in open multi-application smart cards. The key idea lies in the notion of contract, a specification of the security behavior of an application that must be compliant with the security policy of the card hosting the application. In this paper we address a key issue to realize the SxC idea, namely the outsourcing of the contractpolicy matching service to a Trusted Third Party (TTP). In particular, we present the design and implementation of $(SC)^2$ (Secure Communication over Smart Cards), a system securing the communication between a smart card and the TTP which provides the SxC matching service.

Nicola Dragoni, Eduardo Lostal, Davide Papini and Javier Fabra
4th International Symposium on
Foundations & Practice of Security, Paris 2011

Securing Off-Card Contract-Policy Matching in Security-By-Contract for Multi-Application Smart Cards

The Security-by-Contract (SxC) framework has recently been proposed to support applications' evolution in multi-application smart cards. The key idea is based on the notion of contract, a specification of the security behavior of an application that must be compliant with the security policy of a smart card. In this paper we address one of the key features needed to apply the SxC idea to a resource limited device such as a smart card, namely the outsourcing of the contract-policy matching to a Trusted Third Party. The design of the overall system as well as a first implemented prototype are presented.

Nicola Dragoni, Eduardo Lostal, Davide Papini and Javier Fabra
4th International Conference on Mobile Ubiquitous Computing, Systems,
Services and Technologies, Firenze 2010

Bibliography

[1] Atmel CryptoMemory. `https://secure.atmel.com/products/other/securemem/default.aspx`.

[2] IEEE Standard for Reduced-Pin and Enhanced-Functionality Test Access Port and Boundary-Scan Architecture. *IEEE Std 1149.7-2009*, pages 1–985, October 2010.

[3] IEEE Standard for Local and Metropolitan Area Networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). *IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)*, pages 1–314, May 2011.

[4] T. Alpcan and B. Tamer. *Network Security: A Decision and Game Theoretic Approach.* Cambridge Press, 2010.

[5] R. Alur and T. Henzinger. Reactive Modules. *Formal Methods in System Design*, 15(1):7–48, 1999.

[6] F. Bai and A. Helmy. *A Survey of Mobility Modeling and Analysis in Wireless Adhoc Networks.* Springer, Book Chapter in the book "Wireless Ad Hoc and Sensor Networks", 2006.

[7] J. Balasch, B. Gierlichs, R. Verdult, L. Batina, and I. Verbauwhede. Power Analysis of Atmel CryptoMemory - Recovering Keys from Secure EEPROMs. In O. Dunkelman, editor, *Topics in Cryptology - CT-RSA 2012, The Cryptographers' Track at the RSA Conference*, volume 7178 of *Lecture Notes in Computer Science*, pages 19–34, San Francisco, 2012. Springer-Verlag.

[8] H. Baldus, S. Corroy, A. Fazzi, K. Klabunde, and T. Schenk. Human-Centric Connectivity Enabled by Body-Coupled Communications. *Communications Magazine, IEEE*, 47(6):172–178, June 2009.

[9] E. Barker and A. Roginsky. *Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths, NIST SP-800-131A*. National Institute of Standards and Technology, January 2011.

[10] D. Basin, S. Capkun, P. Schaller, and B. Schmidt. Let's Get Physical: Models and Methods for Real-World Security Protocols. In S. Berghofer, T. Nipkow, C. Urban, and M. Wenzel, editors, *Theorem Proving in Higher Order Logics*, volume 5674 of *Lecture Notes in Computer Science*, pages 1–22. Springer Berlin Heidelberg, 2009.

[11] D. Basin, S. Capkun, P. Schaller, and B. Schmidt. Formal Reasoning about Physical Properties of Security Protocols. *ACM Transactions on Information and System Security*, 14(2):1–28, September 2011.

[12] D. Basin and C. Cremers. Degrees of Security : Protocol Guarantees in the Face of Compromising Adversaries. *Computer Science Logic*, pages 1–18, 2010.

[13] M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In D. Stinson, editor, *Advances in Cryptology — CRYPTO' 93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer Berlin Heidelberg, 1994.

[14] Z. Benenson, P. Cholewinski, and F. Freiling. Vulnerabilities and Attacks in Wireless Sensor Networks. *Wireless Sensors Networks Security. IOS Press, Cryptology & Information Security Series*, 2008.

[15] Z. Benenson, A. Dewald, and F. Freiling. Presence, Intervention, Insertion: Unifying Attack and Failure Models in Wireless Sensor Networks. Technical report, University of Mannheim, 2009.

[16] A. Biryukov and I. Nikolić. Automatic Search for Related-Key Differential Characteristics in Byte-Oriented Block Ciphers: Application to AES, Camellia, Khazad and Others. *Advances in Cryptology–EUROCRYPT 2010*, pages 322–344, 2010.

[17] M. Blaze, W. Diffie, R. Rivest, B. Schneier, and T. Shimomura. Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security. A Report by an Ad Hoc Group of Cryptographers and Computer Scientists. Technical report, DTIC Document, 1996.

[18] H. C. Bohnenkamp, P. R. D'Argenio, H. Hermanns, and J.-P. Katoen. Modest: A Compositional Modeling Formalism for Hard and Softly Timed Systems. *IEEE Transactions on Software Engineering*, 32(10):812–830, 2006. ISSN: 0098-5589.

[19] C. Buschmann and D. Pfisterer. iSense: A Modular Hardware and Software Platform for Wireless Sensor Networks. *6. Fachgespräch Sensornetzwerke*, 2007.

[20] D. Cho and C. Chang. Networked Medical Monitoring in the Battlefield. *MILCOM 2008*, pages 1–7, 2008.

[21] S. S. Clark and K. Fu. Recent Results in Computer Security for Medical Devices. In *International ICST Conference on Wireless Mobile Communication and Healthcare*, October 2011.

[22] S. Creese, M. Goldsmith, R. Harrison, B. Roscoe, P. Whittaker, and I. Zakiuddin. Exploiting Empirical Engagement in Authentication Protocol Design. *Security in Pervasive Computing*, pages 119–133, 2005.

[23] S. J. Creese, M. Goldsmith, A. W. Roscoe, and I. Zakiuddin. The Attacker in Ubiquitous Computing Environments: Formalising the Threat Model. In *Formal Aspects of Security*, 2003.

[24] G. De Meulenaer, F. Gosset, F. Standaert, and O. Pereira. On the Energy Cost of Communication and Cryptography in Wireless Sensor Networks. In *Networking and Communications, 2008. WIMOB'08. IEEE International Conference on Wireless and Mobile Computing,*, pages 580–585. IEEE, 2008.

[25] Department for Businnes Innovation and Skills. Cyber Security Guidance for Business. `http://www.bis.gov.uk/policies/business-sectors/cyber-security`, 2012.

[26] A. Di Mauro, D. Papini, and N. Dragoni. Security Challenges for Energy-Harvesting Wireless Sensor Networks. In *Proceedings of the 2nd International Conference on Pervasive Embedded Computing Communication Systems (PECCS 2012)*, Rome, Italy, 2012.

[27] A. Di Mauro, D. Papini, R. Vigo, and N. Dragoni. Introducing the Cyber-Physical Attacker to Energy-Harvesting Wireless Sensor Networks. *Journal of Networking Technology*, 3(3), September 2012.

[28] A. Di Mauro, D. Papini, R. Vigo, and N. Dragoni. Toward a Threat Model for Energy-Harvesting Wireless Sensor Networks. In *Proc. Networked Digital Technologies 4th Internation Conference (NDT 2012)*, pages 289–301, Dubai, April 2012.

[29] D. Dolev and A. Yao. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[30] J. Dunning. Taming the Blue Beast: A Survey of Bluetooth Based Threats. *IEEE Security & Privacy*, 8(2):20–27, March-April 2010.

[31] EMVCo. *EMV Specifications Version 4.3.* EMVCo LLC, `www.emvco.com`, 2011.

[32] V. Forejt, M. Kwiatkowska, G. Norman, and D. Parker. Automated Verification Techniques for Probabilistic Systems. In M. Bernardo and V. Issarny, editors, *Formal Methods for Eternal Networked Software Systems (SFM'11)*, volume 6659 of *LNCS*, pages 53–113. Springer, 2011.

[33] W. Hu, N. Bulusu, C. T. Chou, S. Jha, A. Taylor, and V. N. Tran. Design and Evaluation of a Hybrid Sensor Network for Cane Toad Monitoring. *ACM Transactions on Sensor Networks*, 5(1):4:1–4:28, February 2009.

[34] International Organization for Standardization. *ISO/IEC 7498-1:1994, Open Systems Interconnection Basic Reference Model: The Basic Model, International Organization for Standardization.* Geneva, Switzerland, 1994.

[35] International Organization for Standardization. *ISO/IEC 7816, Identification Cards – Integrated Circuit(s) Cards With Contacts.* Geneva, Switzerland, 1998-2012.

[36] International Organization for Standardization. *ISO/IEC 29192-2:2012, Information Technology – Security Techniques – Lightweight Cryptography – Part 2: Block Ciphers.* Geneva, Switzerland, 2012.

[37] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Advances in Cryptology - CRYPTO'99*, pages 789–789. Springer, 1999.

[38] B. Kordy, S. Mauw, M. Melissen, and P. Schweitzer. Attack-Defense Trees and Two-Player Binary Zero-Sum Extensive Form Games Are Equivalent. *Decision and Game Theory ...*, pages 245–256, 2010.

[39] B. Kordy, S. Mauw, S. Radomirović, and P. Schweitzer. Foundations of Attack-Defense Trees. pages 80–95, 2011.

[40] B. Kordy, S. Mauw, S. Radomirović, and P. Schweitzer. Attack-Defense Trees. *Journal of Logic and Computation*, 2012.

[41] J. Krumm, editor. *Ubiquitous Computing Fundamentals.* Chapman and Hall/CRC, Boca Raton, FL, 2010.

[42] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of Probabilistic Real-time Systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[43] C. Li, A. Raghunathan, and N. Jha. Hijacking an Insulin Pump: Security Attacks and Defenses for a Diabetes Therapy System. In *13th IEEE International Conference on e-Health Networking Applications and Services (Healthcom), 2011*, pages 150–156. IEEE, 2011.

[44] L. Marinos and A. Sfakianakis. ENISA Threat Landscape, Responding to the Evolving Threat Environment. Technical report, 2012.

[45] MATLAB. *version 7.14.0 (R2012a)*. The MathWorks Inc., Natick, Massachusetts, 2012.

[46] N. Mavrogiannopoulos, A. Pashalidis, and B. Preneel. Security Implications in Kerberos by the Introduction of Smart Cards. In *Proceedings of the 7th ACM Symposium on Information, Computer, and Communications Security (ASIACCS 2012)*, page 5, Seoul,KR, 2012. ACM.

[47] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang. Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In K. Paterson, editor, *Advances in Cryptology EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 69–88. Springer Berlin Heidelberg, 2011.

[48] R. M. Needham and M. D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Commununications of the ACM*, 21(12):993–999, December 1978.

[49] V. Nimal. Statistical Approaches for Probabilistic Model Checking. MSc Mini-project Dissertation, Oxford University Computing Laboratory, 2010.

[50] J. Perez, K. Sayrafian-Pour, and J. Geissbuehler. CRAWDAD trace nist/-multihop/experiments/rss_success-rate (v. 2007-08-20). Downloaded from `http://crawdad.cs.dartmouth.edu/nist/multihop/experiments/rss_success-rate`, August 2007.

[51] N. Potlapally, S. Ravi, A. Raghunathan, and N. Jha. A Study of the Energy Consumption Characteristics of Cryptographic Algorithms and Security Protocols. *IEEE Transactions on Mobile Computing*, 5(2):128–143, 2006.

[52] P. Rothenpieler, D. Krüger, D. Pfisterer, S. Fischer, D. Dudek, C. Haas, A. Kuntz, M. Zitterbart, C. Buschmann, C. Wieschebrink, et al. Flegsens-Secure Area Monitoring Using Wireless Sensor Networks. *Proceedings of the 4th Safety and Security Systems in Europe*, 2009.

[53] Royal Society. *Risk: Analysis, Perception and Management - Report of a Royal Society Study Group*. 1992.

[54] B. Schneier. Attack Trees. *Dr. Dobbś journal*, 24(12):21–29, 1999.

[55] V. Shoup and A. Rubin. Session Key Distribution Using Smart Cards. In *Advances in Cryptology-EUROCRYPT96*, pages 321–331. Springer, 1996.

[56] D. Singelée, S. Seys, L. Batina, and I. Verbauwhede. The Communication and Computation Cost of Wireless Security - Extended Abstract. In *Proceedings of the 4th ACM conference on Wireless network security (WiSec 2011)*, pages 1–3, Hamburg,GER, 2011. ACM.

[57] N. Smart. ECRYPT II Yearly Report on Algorithms and Keysizes (2010-2011). Technical report, ECRYPT II - European Network of Excellence in Cryptology II, 2011. ICT-2007-216676 `http://www.ecrypt.eu.org/documents/D.SPA.17.pdf`.

[58] B. Sullivan. Exclusive: Millions of Printers Open to Devastating Hack Attack, Researchers Say. `http://redtape.msnbc.msn.com/_news/2011/11/29/9076395-exclusive-millions-of-printers-open-to-devastating-hack-attack-researchers-say`, 2011.

[59] Z. Sun, P. Wang, M. C. Vuran, M. A. Al-Rodhaan, A. M. Al-Dhelaan, and I. F. Akyildiz. BorderSense: Border Patrol Through Advanced Wireless Sensor Networks. *Ad Hoc Networks*, 9(3):468–477, 2011.

[60] A. V. Taddeo, M. Mura, and A. Ferrante. QoS and Security in Energy-Harvesting Wireless Sensor Networks. In *Security and Cryptography (SECRYPT), Proceedings of the 2010 International Conference on*, pages 1–10, July 2010.

[61] E. Tews, R. Weinmann, and A. Pyshkin. Breaking 104 bit WEP in Less than 60 Seconds. *Information Security Applications*, pages 188–202, 2007.

[62] R. Want, B. Schilit, N. Adams, R. Gold, K. Petersen, D. Goldberg, J. Ellis, and M. Weiser. An Overview of the PARCTAB Ubiquitous Computing Experiment. *Personal Communications, IEEE*, 2(6):28 –43, December 1995.

[63] M. Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):94–104, 1991.

[64] M. Weiser. Hot Topics-Ubiquitous Computing. *Computer, IEEE*, 26(10):71–72, 1993.

[65] M. Weiser. Some Computer Science Issues in Ubiquitous Computing. *Communications of the ACM*, 36(7):75–84, 1993.