Modelling and Quantitative Analysis of Performance vs Security Trade-offs in Computer Networks

Esmaeil HABIB ZADEH

PhD

University of Bradford

2017

Modelling and Quantitative Analysis of Performance vs Security Trade-offs in Computer Networks

An investigation into the modelling and discrete-event simulation analysis of performance vs security trade-offs in computer networks, based on combined metrics and stochastic activity networks (SANs)

Esmaeil HABIB ZADEH

Submitted for the Degree of

Doctor of Philosophy

Faculty of Engineering and Informatics

University of Bradford

2017

Abstract

Esmaeil Habib Zadeh

Modelling and Quantitative Analysis of Performance vs Security Trade-offs in Computer Networks

An investigation into the modelling and discrete-event simulation analysis of performance vs security trade-offs in computer networks, based on combined metrics and stochastic activity networks (SANs)

Keywords: Performance, security, trade-off, modelling, simulation, petri nets, stochastic activity networks

Performance modelling and evaluation has long been considered of paramount importance to computer networks from design through development, tuning and upgrading. These networks, however, have evolved significantly since their first introduction a few decades ago. The Ubiquitous Web in particular with fast-emerging unprecedented services has become an integral part of everyday life. However, this all is coming at the cost of substantially increased security risks. Hence cybercrime is now a pervasive threat for today's internet-dependent societies. Given the frequency and variety of attacks as well as the threat of new, more sophisticated and destructive future attacks, security has become more prevalent and mounting concern in the design and management of computer networks. Therefore equally important if not more so is ***security***.

Unfortunately, there is no one-size-fits-all solution to security challenges. One security defence system can only help to battle against a certain class of security threats. For overall security, a holistic approach including both reactive and proactive security measures is commonly suggested. As such, network security may have to combine multiple layers of defence at the edge and in the network and in its constituent individual nodes.

Performance and security, however, are inextricably intertwined as security measures require considerable amounts of computational resources to execute. Moreover, in the absence of appropriate security measures, frequent security failures are likely to occur, which may catastrophically affect network performance, not to mention serious data breaches among many other security related risks.

In this thesis, we study optimisation problems for the trade-offs between performance and security as they exist between performance and dependability. While performance metrics are widely studied and well-established, those of security are rarely defined in a strict mathematical sense. We therefore aim to conceptualise and formulate security by analogy with dependability so that, like performance, it can be modelled and quantified.

Having employed a stochastic modelling formalism, we propose a new model for a single node of a generic computer network that is subject to various security threats. We believe this nodal model captures both performance and security aspects of a computer node more realistically, in particular the intertwinements between them. We adopt a simulation-based modelling approach in order to identify, on the basis of combined metrics, optimal trade-offs between performance and security and facilitate more sophisticated trade-off optimisation studies in the field.

We realise that system parameters can be found that optimise these abstract combined metrics, while they are optimal neither for performance nor for security individually. Based on the proposed simulation modelling framework, credible numerical experiments are carried out, indicating the scope for further work extensions for a systematic performance vs security tuning of computer networks.

Declaration

The candidate confirms that the work submitted is his own and that appropriate credit has been given where reference has been made to the work of others.

Esmaeil Habib Zadeh

*To my wife, **Shadi**, and our son, **Pedram***

Acknowledgements

First and foremost, I thank God for giving me the patience, strength, ability and willingness to keep learning and the opportunity to undertake this research study and persevere and complete it satisfactorily. Without his blessings, this achievement would not have been possible.

My best appreciation goes to my supervisor, Prof. Demetres D Kouvatsos, for all his support, advice and guidance during my entire study; and for his patience, understanding and generosity so that I could survive as a full-time student while fulfilling my other work and family commitments.

Lastly, but by no means lesser in importance, I sincerely express my appreciation to my wife, Shadi, and our son, Pedram, who have been fully understanding and supportive throughout this long journey; without their love and patience, I could not be where I am now. I also would like to extend my deepest sense of gratitude towards all our families and friends for their unconditional support and affection, which have been tremendous sources of strength, energy and relief for all of us throughout.

Publications and Presentations

Workshops

Habib Zadeh E., Kouvatsos D. D., and Miskeen G. M. A., A Hybrid Simulation Framework for the Analysis of Petri Nets (PNs) and Queueing Networks (QNs), UKPEW 14, Newcastle University, UK, 2014

Kouvatsos D. D., Habib Zadeh E., and Miskeen G. M. A., (2014). Performance Modelling and Evaluation of Secure Dynamic Group Communication Systems in RANETs, UKPEW 14, Newcastle University, UK, 2014

Habib Zadeh E., Performance Implications of Intrusion Detection and Prevention Systems (IDPSs), Networks and Performance-Security Engineering (NetPSEn), University of Bradford, UK, 2015

Tutorials

Habib Zadeh E., Kouvatsos D. D. and Miskeen G. M. A., Hybrid Petri Queueing Networks Simulator (HPQNS), HET-NETs 2013 Conference, 11th - 13th of November 2013, Ilkely, Bradford, UK.

Miskeen G. M. A., Kouvatsos D. D. and Habib Zadeh E., Performance vs Security Trade-offs in RANETs Based on Hybrid Quantitative Network Models, HET-NETs 2013 Conference, 11th - 13th of November 2013, Ilkely, Bradford, UK.

Journals

Miskeen G. M. A., Kouvatsos D. D., and Habib Zadeh E., (2013). An Exposition of Performance-Security Trade-offs in RANETs Based on Quantitative Network Models, Wireless Personal Communications, 2013, Vol. 70, Issue 3, June 2013, 1121-1146, DOI 10.1007/s11277-013-1105-0.

Work in progress

Habib Zadeh E., Kouvatsos D. D., (2018). Nodal Model-based Trade-offs between Performance and Security in Computer Networks

Table of Contents

List of Figures

List of Tables

Acronyms

AFI	Abstract Functional Interface
BS	Buffer Size
CERT	Computer Emergency Response Team
CIS	The Center for Internet Security
CTMC	Continuous Time Markov Chain
CU	Compromised Undetected
DDoS	Distributed Denial of Service
DoS	Denial of Service
FA	False Alarm
FCFS	First Come First Served
FN	False Negative
FP	False Positive
GSPN	Generalised Stochastic Petri Net
HIDS	Host-based Intrusion Detection System
HOL	Head-of-Line
IADS	Insider Attack Detection Sub-model
IDPS	Intrusion Detection and Prevention System
IDRS	Intrusion Detection and Response System
IDS	Intrusion Detection System
IPS	Intrusion Prediction System
ISSTD	Insiders Security State Transition Diagram
LAN	Local Area Network
MANET	Mobile Ad Hoc Network
MCMV	Mean Cost to Mitigate Vulnerabilities
MCOI	Mean Cost of Incident
MCP	Mean Cost to Patch
MET	Mean Encryption Time
MIAT	Mean Inter-Arrival Time
MIIT	Mean Inter-Inspection Time
MIRC	Mean Incident Recovery Cost
MIT	Mean Inspection Time
MTBF	Mean Time Between Failures
MTBSI	Mean Time Between Security Incidents
MTIR	Mean Time to Incident Recovery
MTT	Mean Transmission Time
MTTIA	Mean Time To Insider Attack
MTTID	Mean Time to Incident Discovery
MTTIV	Mean Time To Insider Vulnerability
MTTMV	Mean Time to Mitigate Vulnerabilities
MTTOA	Mean Time To Outsider Attack
MTTOV	Mean Time to Outsider Vulnerability

MTTP	Mean Time to Patch
MTTSF	Mean Time To Security Failure
MTTVA	Mean Time To Virus Attack
MTTVV	Mean Time To Virus Vulnerability
NIDS	Network-based Intrusion Detection System
NOI	Number of Incident
OADS	Outsider Attack Detection Sub-model
OSSTD	Outsiders Security State Transition Diagram
PEPA	Performance Evaluation Process Algebra
PLP	Packet Loss Probability
PN	Petri Net
QN	Queueing Network
QoS	Quality-of-Service
RANET	Robot Mobile Ad Hoc Network
RRE	Restoration Reconfiguration Evolution
SAN	Stochastic Activity Network
SGCS	Secure Group Communication System
SMC	Semi-Markov Chain
SPN	Stochastic Petri Net
SSS	Shared Server Scheduler
SSTD	Security State Transition Diagram
TOEP	Transmission Over Encryption Priority
TP	True Positive
VADS	Virus Attack Detection Sub-model
VDP	Vulnerability Detection Probability
VSSTD	Virus Security State Transition Diagram
WEP	Wired Equivalent Privacy

Chapter 1

Introduction

In this opening chapter, we provide a brief overview about the two crucially important aspects of computer networks - *performance* and *security*. Our discussions in particular concentrate on the interrelationships between performance and security and the legitimate requirements for model-based analysis of trade-offs between them. We then review the motivations for this thesis and restate the problems we aim to address through this research study. At the end, we provide a brief list of contributions of this thesis as well as its structure.

1.1 Performance in Computer Networks

Computer networks are always designed with required Quality-of-Service (QoS) parameters, also known as Service Level Agreements (SLAs), that may well be expressed in terms of some commonly used performance metrics such as *throughput*, *end-to-end delay*, *packet loss probability* and *utilisation* of certain components, just to name a few. Performance modelling and evaluation plays an important role in the design and management of computer networks providing flexible and dependable tools which can for instance be utilised for capacity planning, optimisation and prediction analysis. Such tools allow for early assessment of critical components and/or potential issues so as to guarantee the best cost-performant systems [40, 50, 85, 86].

Nevertheless, it is not uncommon to conduct such performance evaluations near completion of the systems. At that late stage, it might be necessary to undertake a major redesign to correct serious performance problems, which may lead to expensive, inefficient, time consuming and professionally irresponsible system designs [65].

There are three basic techniques which can be employed to conduct performance evaluation in computer networks: *analytical modelling*, *simulation*, and *measurement* [11, 40]. Depending on certain consideration factors, one of these might be the most preferable and/or the only feasible technique to be adopted. Jain [40] provides a list of such considerations (from most to least important) which may help decide when to use which technique(s) as follows: *stage*, *time required*, *tools*, *accuracy*, *trade-off evaluation*, *cost* and *saleability.*

For instance, *analytical modelling* and *simulation* would be the only methods to be considered if the network did not exist in the first place, or if the system existed but it would be enormously costly to interrupt it in order to take some required measurements. In general, however, any *analytical modelling* or *simulation* would be more convincing if they were carried out based on some previously captured actual measurements [40].

In the context of this thesis, *accuracy* too is of considerable importance; with this in mind, below comes a brief comparison between *analytical modelling* and *simulation* solutions.

Analytical modelling is to represent the formal system description as some mathematical formulas which may provide exact or approximate solutions. While *analytical models* are quite fast and cost-effective, there are no feasible analytical solutions for many real-life systems [12, 85]. Abstracting such complex systems would require too much simplification with unrealistic assumptions, potentially leading to models that at best will not generalise well.

Simulations, on the other hand, require much less underlying assumptions and can flexibly mimic any behaviour of the system under consideration in various levels of details as appropriate [85]. Hence *simulation* methods, as expected, provide results much closer to reality. This may, however, come at significant cost in terms of increased run times and higher difficulty levels in optimisation studies [12, 40].

1.2 Security in Computer Networks

The Ubiquitous Web with fast-emerging unprecedented services has, unfortunately, come at substantially increased security risks, potentially affecting *every* individual regardless. Cybercrime is now a pervasive threat for today's internet-dependent societies [84]. Given the frequency and variety of attacks as well as the threat of new, more sophisticated and destructive future attacks, network security has become a central topic [50]. Data privacy in particular has gained a lot of attention.

To battle against all these security threats, network security has developed remarkably. It involves any activity, in terms of both hardware and software, designed to protect the usability and integrity of the network and the data. It may include various solutions each of which aiming to protect against a certain class of threats. Network security therefore may combine multiple layers of defence at the edge and in the network and in its constituent individual nodes, where each layer implements certain policies and controls to grant access to authorized users but, at the same time, deny any access to malicious actors [27]. The ultimate goal of security, as widely agreed, [45, 50, 52], is to protect three unique attributes of data - *confidentiality*, *integrity* and *availability*.

More specifically, *confidentiality* is about ensuring that data can only be accessed or understood by authorised persons, which necessarily requires the employment of cryptographic techniques (encryption/decryption) to prevent unauthorised access. *Integrity* concerns about the maintenance of data over its life-cycle and that data cannot be altered by unauthorised parties, either maliciously or by accident. And equally important is *availability* that ensures data being readily accessible to authorised users when needed.

To enforce these attributes, it is not uncommon to recommend taking a holistic approach including both *reactive*(e.g. vulnerability patching) and *proactive* (e.g. cryptography, firewalls) security measures[1]. However, performance and security are inextricably intertwined as security measures require significant amounts of computational resources to execute. Moreover, in the absence of appropriate security measures, frequent security failures are likely to occur, which may potentially lead to catastrophic network performance degradation as well as serious data breaches. Below is, by no means an exhaustive list but, just to give an indication of some security measures widely employed together with their potential impact on performance.

- As the first line of defence, organisations usually have firewalls installed which, if properly configured, can carefully scrutinise ***any*** access to the network resources. Although, from security perspective, this may be a highly effective solution, firewalls can become serious bottlenecks, leading to significant performance degradation in the networks [50, 57, 87].

- The employment of encryption/decryption mechanisms in order to protect data against outsider attacks is also a most-favoured solution. Despite proven costly from computational standpoint [51, 57, 64, 65], these procedures are implemented such that several network security protocols with large degrees of overlap may in fact happen to be running simultaneously. As a result, encryption/decryption may impose considerable delays on data communications across the networks.

- As computer networks are becoming ever more complex, there are always exploitable weaknesses such as design and programming errors as well as various socially engineered penetration techniques [114]. It is, therefore, widely accepted that prevention mechanisms alone are no longer sufficient to provide the desired protection for threatened networks. It is essential to also consider Intrusion Detection System (IDS)[2] as yet another line of defence.

[1]Security measures are defined as measures taken as a precaution against theft or espionage or sabotage etc. More specifically, security measures here refer to such countermeasures as anti-viruses, Intrusion Detection and Response Systems (IDRSs), Encryption/Decryption mechanisms, etc. that can be employed to battle against security attacks.

[2]In this thesis, the terms IDS, Intrusion Detection and Prevention System (IDPS) and Intrusion

IDS are designed so that they can help with early detection of malicious activities and, therefore, properly respond to remedy or mitigate the potential damages intended [24, 60, 114]. Furthermore, IDS may be used in the network level and/or in the host level, respectively called *Network-based Intrusion Detection System (NIDS)* and *Host-based Intrusion Detection System (HIDS).*

- Despite substantial advances in security countermeasures, attacks and intrusions continue to occur, hence the inefficacy of reactive and proactive techniques [1, 2]. Recent developments in data storage, computation and analytics are now inspiring the security research community to adopt the next generation of security defence tools by seriously considering the Intrusion Prediction System (IPS) [1].

As expected, such security measures as IPS and IDS involve far more in depth inspection and analysis of traffic, user profile/activity. While these exhibit more effective and promising solutions to the future security threats [1, 2, 91], they are computationally expensive and highly likely to pose significant implications on performance of the networks.

1.3 Performance vs Security Trade-off

A trade-off is a balance between two opposing things, where to achieve one thing we would have to pay in terms of the other. Recall that security measures are employed to improve security; however, they come at the cost of performance. Lifting security measures should supposedly result in some boosts in performance, but in reality this can lead to less secure systems and hence more prone to security failures. Such systems are likely to incur catastrophic performance degradation, let alone the implications for security such as data breaches. Therefore, trade-off studies can help identify optimal system parameters that ensure best balance between performance and security.

Detection and Response System (IDRS) are interchangeably used to refer to a class of security measures that have the capability to detect, fix and respond to security threats.

Trade-off analysis, for instance, is extensively carried out in security key distribution centres [117, 118] to optimise new key generation intervals with respect to the cost incurred by the network. According to Stallings [98], *"The more frequently session keys are exchanged, the more secure they are, because the opponent has less ciphertext to work with for any given session key. On the other hand, the distribution of session keys delays the start of any exchange and places a burden on network capacity. A security manager must try to balance these competing considerations in determining the lifetime of a particular session key."*

Nevertheless, it is well-understood that different systems would have different performance and security requirements to ensure best cost-performant designs. Sometimes there exist extremes with respect to either performance or security where the solution of choosing between them might seem obvious. For instance, in embedded systems with low computational power, strict energy constraints and little outside interaction, security would be of little concern and the focus, thus, typically should lie on performance and energy-efficiency. At the other end are enterprise systems with extremely high security requirements as well as the need for sufficient computing facilities. Here, higher security levels are mostly favoured and the cost in terms of performance is readily accepted, although this is usually compensated for by provision of more (powerful) resources.

For systems between these extremes, however, the choice of optimum may not be so apparent where a model-based approach can be useful [112]. The adoption of a model-based approach can allow the analysts to a) obtain useful insights into the system from the early phases of a system design, and b) perform "what-if" analyses in order to estimate the impact deriving from architectural changes [72]. Miskeen et al. [70, 71] recommend that an optimal performance and security trade-offs modelling and evaluation approach should be considered through the design, development, tuning and upgrading of computer networks in order to prove most effective.

1.4 Research Motivation and Problem Statement

We realise that performance and security are closely interrelated and that there often exist trade-offs between them to be achieved. Most existing work, however, only focuses on either *performance* or *security*, without paying much attention to the profound implications one may have for the other.

We also note that while qualitative security has been quite widely investigated, only little attention has been paid for quantification of security, of which mostly is done by analogy with dependability. For instance, analogous to *availability* in the world of dependable computing is the probability of the network being secure; or similar to the Mean Time Between Failures (MTBF) metric in reliability is the Mean Time To Security Failure (MTTSF) [18, 43, 61, 75, 106, 112]. Nevertheless, model-based quantitative techniques are still rare and in their infancy. We are, therefore, motivated to develop stochastic quantitative modelling techniques to investigate optimal trade-offs between performance and security.

However, any attempt at performance vs security trade-off analysis implies that both performance and security can be quantitatively measured. While performance metrics have been extensively studied and are well-established, that is not quite the case for security.

Analogous to dependability, we therefore aim to conceptualise and formulate security so that, like performance, it can be modelled and quantified. We then wish to formalise the optimisation problems for the trade-offs between performance and security as they exist between performance and dependability. We adopt a simulation-based modelling approach, for its generality and flexibility, in order to identify optimal trade-offs between performance and security for a single node of a generic computer network, subject to a number of security threats. Based on a stochastic modelling formalism, we propose a new nodal model which can capture more realistically both performance and security aspects, in particular the intertwinements between them.

Our empirical approach, we believe, utilises optimal combined metrics of both performance and security and facilitates more sophisticated optimisation studies in the field. We note that system parameters can be found that optimise these abstract measures, while they do not optimise performance nor security individually. Based on the proposed simulation modelling framework, credible numerical experiments are carried out, indicating the scope for further work extensions for a systematic performance vs security tuning of computer networks.

1.5 Research Contributions

We aim to study optimisation problems for the trade-offs between performance and security as they exist between performance and dependability. By analogy with performability modelling, where one combines a performance model and a dependability model with the goal of jointly evaluating performance and dependability as well as the dependencies between parameters controlling either, here we aim at jointly modelling and evaluating performance and security, in particular their intertwinments as well as the implications of one for the other.

It is fair to claim that all chapters of this research thesis could be seen, in one way or another, as a contribution to the research community.

As seen, Chapter 1 sets the scene and describes the importance of performance and security and the interrelationships between them in today's Internet dependent societies. In this chapter, we also attempt to explain the requirements for optimisation of trade-offs between performance and security, in particular why utilisation of model-based quantitative approaches should be considered.

In Chapter 2, we provide a summary of our extensive exploration of the existing work with regard to the thesis topic. The performance implications of both security threats and security measures have been our main focus during this thorough search. More specifically, we summarise, based on literature, how security measures are computationally expensive, hence they are highly likely to impose significant overheads on the networks. We also review the literature for the existing approaches about modelling and quantification of security and the challenges ahead, where we compare and contrast the existing methodologies. More importantly, we critically review and evaluate the trade-off studies, in particular those of performance vs security. We realise that there is little amount of work which concerns model-based quantitative analysis of trade-offs. We discuss and summarise the advantages and disadvantages of these approaches as opposed to our approach exercised in this research work.

In Chapter 3, we attempt to conceptualise and formulate security so that, like performance, it can be modelled and quantified. Our contribution to this chapter is three-fold:

- While most of our work for the state transition diagram introduced in this chapter is derived from the existing work, we propose a security state transition diagram which forms the basis for a combined complementary security solutions. More specifically, we are determined to explicitly incorporate both false positive (FP) and false negative (FN) alarm signals in our model. This not only helps provide a more realistic representation of the system under consideration, but it also enhances capability of the model so that further performance and security metrics that used to be hidden can now be monitored and collected. As a result, our model can be used to investigate the costs in terms of performance and security associated with each of these alarm signals. It is worth mentioning that false alarms are considered inherent characteristics of security measures, where FN signals can lead to serious data breaches and FP signals can affect the availability and performance of the systems.

- We also conceptualise and formulate the behaviours of the attackers and those of the security measures so that their implications for performance can be modelled and quantified.

- To measure and report security quantitatively, we review the literature and put together a standardised list of security metrics which are utilised throughout this thesis.

In Chapter 4, comes the main contribution of this research work. Here, we propose a stochastic model-based approach that enables quantitative analysis of combined performance and security so that optimisation studies can be carried out for trade-offs between them. Due to our generic and modular approach in design, the proposed model may be employed to study a range of computer nodes with varying security concerns and/or mission priorities. In particular, it allows for various modules to be dynamically added or removed to suit different application areas where either security, performance, energy or combinations of these are of concern. We further discuss how different modules could be combined to provide an integrated model solution facilitating quantitative analysis of real life systems. In addition, we introduce and describe some combined abstract measures that are used to analyse and optimise the trade-offs between performance and security. We realise that system parameters can be found so as to optimise these combined metrics, while such parameters would optimise neither performance nor security individually.

In Chapter 5, we discuss the results obtained from some numerical experiments and provide physical interpretations. Through various simulation scenarios, we aim at highlighting security related implications for performance. More specifically, we investigate the impacts of different security key lengths on performance, security and the trade-offs between them. We realise that system performance can be dramatically affected if we chose very short or very long security keys, although for different reasons. As discussed in Chapter 4, such optimal trade-offs between performance and security are achieved for certain system parameters. Moreover, we conduct similar simulation runs to demonstrate the significance of false alarm signals and their relation with periodic inspection intervals of security measures such as anti-viruses and/or Intrusion Detection Systems (IDSs). Unlike the common belief that security measures should be executed as often as possible, we find that they should be performed at optimal rates to enhance overall system security. This is due to the fact that having non-zero false alarm rates is an inherent characteristic of security measures.

Last but not least, in Chapter 6, we summaries possible extensions to this work together with some open questions and challenges that we have identified throughout the whole study. In particular, the preliminary analysis of cost and energy submodels has been indeed intriguing in spite of the fact that we did not manage to complete our intended work with regard to the **Cost Functions** and **Energy Consumption** sections due to time constraints. We therefore plan to carry out the work on this in the hope that we can further contribute to the field. Other possible directions for future work are also listed and explained in reasonable details.

1.6 Thesis Structure

The rest of this thesis is organised as follows.

Chapter 2 extensively explores the literature for a series of relevant topic areas; in particular three classes of commonly employed security attacks and their respective security measures are reviewed with the focus on whether and how these security related tasks could have any significant impact on performance of computer networks. We also investigate the existing methodologies with the focus on stochastic model-based approaches, where attempts are made to quantitatively examine security aspects of computer networks. More importantly, we examine the trade-off studies where performance and security are traded off against each other.

In Chapter 3, we make attempts to conceptualise and formulate security. We propose a security state transition diagram model which we later transform into a stochastic model that helps quantify the security aspects in an individual node. Moreover, we formulate behaviour of attackers and security defence systems. The tasks initiated by security measures are translated into and reflected as implications for performance. In order to measure security in a more consistent manner, we collect and explain in brief a list of security metrics, we use a standardised list of metrics throughout this research for consistency.

In Chapter 4, we propose a stochastic model-based design that enables quantitative analyse of not only performance and security individually, but also the interrelationships between them; due to its generic and modular approach, the model may be employed to study a range of computer nodes with varying security concerns and/or mission priorities. More specifically, the model can be tailor-made to be used in application areas where either security, performance, energy or combinations of these are of concern. We further discuss various modules that are combined to provide an integrated model solution facilitating quantitative analysis of real life systems. The chapter also introduces some combined abstract measures that can be used in trade-offs between performance and security.

In Chapter 5, for the sake of demonstration, we briefly visualise and conduct preliminary analysis of some numerical experiments conducted. We provide physical interpretations for the results obtained. More specifically, we look into the effects of different security key lengths on trade-offs between performance and security. In addition, the impacts of False Alarm (FA) and the time interval between two successive executions of security measures on various measures are investigated.

Chapter 6 summarises our research thesis and outlines some future directions and research areas that could be extended on the basis of the findings of the current thesis.

Chapter 2

Literature Review

Various aspects of security and performance have been extensively studied in literature. In this chapter, we survey the work most relevant to the modelling and quantification of both security and performance, in particular the trade-offs between them. In Section 2.1, we briefly review the existing work with regard to some typical security attacks and the most common practices employed to confront them. The focus of this review is mainly on performance implications of security measures. In Section 2.2, we draw the attention to the complexities with regard to quantitative analysis of security. We provide an overview of existing work that involves security modelling and quantification and the challenges ahead. Section 2.3 explores literature to summarise the state of the art in trade-off analysis and optimisation involving both performance and security. We compare and contrast these methods and briefly highlight the contributions made by our research work.

2.1 Performance Implications of Security Measures

In this section, we survey the existing research with regard to security measures and the profound implications they pose for performance. An exhaustive review of security attacks and their respective countermeasures is beyond the scope of this thesis. Here we only consider three different categories of security threats. More specifically, we focus our attention on the attacks originated by *malware*, the *insiders* and the *outsiders* and explore the common practices for mitigating their destructive impacts. Moreover, the overall performance implications of such remedial measures are reviewed.

Computer networks are susceptible to malicious attacks which may take various forms each of which acting in their own unique ways [47]. To ensure effective and holistic defence strategy, a combination of security measures are required to be in place [44] based on which, a security process may be well-grouped into three distinct phases: *prevention*, *detection* and *response* [24, 52].

2.1.1 Preventative Approach

Deployment of firewalls, encryption and decryption mechanisms, and anti-malware applications are most common preventative approaches to secure computer networks in the first place. Below we discuss each in further details as follows.

Firewalls

As the first line of defence, firewalls are installed so that any access to the network resources can be carefully scrutinised. A firewall may be a piece of software, hardware or combination of both that isolates an organization's internal network from the outside world. Firewalls, usually rule-based, are responsible to security-check, log, drop or forward any traffic flow to/from the networks. It is, therefore, crucial that they themselves are immune to penetration as otherwise they can be compromised, thereby providing only a false sense of security [50]. Depending on security policies and, thus, level of inspections, firewalls may become a serious bottleneck, leading to significant performance implications in the network [50, 57, 87].

Salah et al. [87] investigate the performance of a firewall under Distributed Denial of Service (DDoS) attacks. They point out that poorly designed firewalls can jeopardise the overall security and performance of a network. Using a queueing paradigm, they develop an analytical model to study and compare performance of a firewall in the absence/presence of DDoS attacks. Performance metrics of interest include throughput, packet loss, packet delay and utilisation of the firewall; the overall performance impact is shown to be significant, regardless. In addition, it is illustrated that, under specific rule settings, the firewall becomes unable (fully utilised) to digest the introduced traffic, thereby leading to an unacceptable condition where over 80% of the arriving packets are lost! They use simulation and experimental measurements to verify and validate the outcomes from the analytic models.

Xu and Su [113] conduct a similar performance-based study with the focus on comparisons of two mainstream firewall solutions, a hardware-based (Cisco ASA 5505) and a software-based (Linux iptables). They consider three different types of traffic: a) fixed packet length with no burst; b) random packet length with no burst; and c) random packet length with burst. While the general trends are in agreement with [87], the results also indicate that bursty traffic impacts on the performance more than others, as expected. The paper concludes that the performance implication of firewalls is not only determined by how advanced the hardware is, but also lies on an optimised algorithm employed. The study provides interesting insights into the two types of firewalls which are beyond the scope of this thesis.

Similar studies of firewalls and their impact on performance are conducted by Sheth and Thakker [93] and Funke et al. [34]. Here firewalls are configured differently in order to provide different levels of security. For given security levels, performance implications are examined and visualised for comparison.

While the focus of these works is to compare and contrast the profound impact of security measures (firewalls) on performance, they do not carry out any trade-offs between them.

Encryption/Decryption

Data encryption/decryption has been the long-standing answer to security and authentication concerns [96]. They are widely employed to reduce threats from outsider attacks [24]. While encryption algorithms come in a variety of types and versions, they are all very expensive from computational standpoint. They are likely to impose major overheads on computer networks, regardless, although the amount of overhead may well depend on various parameters such as the security key length and the rate at which packets are exchanged [64, 65].

Performance implications of encryption algorithms have been extensively investigated [32, 51, 95, 104]. Security is not measured here but rather assumed to be an indirect input that can be controlled by other input parameters such as input data length, security key length, security mode, type of algorithm, etc. Given a security level achieved for a certain set of security parameters, the resulting performance in terms of encryption time, resource utilisation, memory usage and energy consumption is measured and interpreted.

In the following, we briefly discuss the work by Lamprecht et al. [51] as it appears to have covered a wider range of encryption algorithms and the techniques employing them. It is worth mentioning, however, that while other researchers have as well considered other metrics such as energy consumption, resource utilisation, memory usage [32], Lamprecht et al. [51] have solely looked into the encryption times; nevertheless, that is not a major constraint since encryption times associated with security key lengths are of most interest to our research in this thesis.

Lamprecht et al. [51] conduct a comprehensive research on the performance of encryption algorithms. Their research involves well-known cryptographic techniques including *symmetric*, *asymmetric* and *hashing*, which are widely employed in online secure transactions to ensure message confidentiality, message integrity, non-repudiation and sender authentication. The authors further consider the most common cryptographic algorithms for each of these techniques to determine their suitability for systems with real-time constraints. Lamprecht et al. carry out experiments involving different combinations of encryption techniques and encryption algorithms versus various message sizes, key lengths (modulus sizes), security modes as well as implementations. As pointed out earlier, the time taken by encryption algorithms is the only performance metric used to evaluate and compare these security algorithms.

Despite slight differences in scope, results and presentations, the overall outcome of all these studies is in agreement with the fact that encryption/decryption algorithms are computationally intensive processes and may pose a considerable amount of overhead. In addition, the results suggest that the time required for such processes is directly proportional to the security key length; that is, the longer the security key, the higher the encryption time, hence the higher the performance impact. Such a relation between key lengths and run-times can be translated into the trade-offs between security and performance which appears to be beyond the scope of these studies.

Modelling performance and reliability of a secure electronic voting scheme is considered by Thomas [105], where encryption and decryption of data is a dominant process. As argued in the paper, to satisfy security requirements of such voting schemes, the network communications are dramatically increased. Insights from modelling and numerical analysis of collected data confirm that the sheer increase of encryption/decryption overhead with respect to the number of voters participated lies at the root of substantially degraded performance.

Kouvatsos and Miskeen [49] investigate performance implications of the Wired Equivalent Privacy (WEP) security protocol in Robot Mobile Ad Hoc Network (RANET). Simulation results clearly show that activation of the WEP protocol has an adverse effect on performance of RANET, with both First Come First Served (FCFS) and Head-of-Line (HOL) queueing disciplines. Further experimental analysis confirms that performing encryption/decryption algorithms and associated computation iterations under WEP may cause an unacceptable increase in the overall packet delay as well as excessive power consumption within RANET.

These findings can help to quantify the cost of security mechanisms, in particular encryption/decryption, in terms of performance.

Anti-malware

Malware is a general term for various malicious code (malcode), most commonly used in the forms of viruses and worms [26]. The level of damage caused by malware can range from minor irritation to stealing confidential information, to destroying data [15, 26, 82].

Of important interest to this thesis is that tools are commonly designed to fight one type of malware and they are usually unable to effectively eradicate threats of other types despite similarities in features and characteristics. For instance, anti-virus software may not be able to detect spyware or email worms [44, 82]. Reavis [82] emphasises the importance of malware attacks and suggests a holistic approach. While such an approach may help significantly improve security of the system under consideration, it would not always be an acceptable solution due to its substantial costs in terms of performance.

Uluski et al. [107] investigate the impact of anti-virus applications on performance. They choose four commonly used anti-virus packages and try to characterise the workloads and their respective overhead imposed on the system by individual execution of them through on-access scanning. The results provided show the significance of performance implications.

It is well worth mentioning that a typical anti-virus program may also be scheduled to work in an on-demand mode, where they are likely to demonstrate noticeably different characteristics and overhead than those of the on-access scanning mode. That said, while the ultimate level of overhead may well depend on the configurations and user activity levels, the overall impacts of anti-viruses on performance should not be underestimated.

Equally important is FA rate, one of inherent characteristics of all security measures, in general, and anti-virus applications, in particular; that is, the rate at which they are likely to issue false signals. In other words, how likely they are to falsely identify normal activities performed by legitimate users as a virus attack (False Positive (FP)); or how often a typical anti-virus program may fail to detect actual virus threats (False Negative (FN)). These rates may meaningfully affect both performance (FP) and security (FN). It is therefore of great importance to investigate these inherent features of anti-virus tools as well when evaluating their effectiveness and performance.

Anti-virus packages traditionally adopted a re-active (signature based) approach to malicious code [77, 80, 101, 107] where signatures would be typically developed in response to known threats. Although such approaches very often detect viruses of known signatures quickly and accurately (with very low FP rates), they are more prone to issue FN signals due to their lack of knowledge about newly generated/distributed viruses. The trend, however, appears to be changing as modern approaches are increasingly utilising a combination of advanced heuristic [8] and behavioural based technologies [111] in addition to non-heuristic (traditional) signatures.

It goes without saying that modern anti-viruses utilising combined methods are likely to demonstrate much improved overall detection rates as opposed to their predecessors. However, the incurred cost in terms of performance is twofold: a) they are highly likely to pose far more overhead due to increased computational processes, and that b) they are equally prone to issue FP signals too, leading to increased probability of the network being unavailable.

It is, once again, emphasised that anti-virus applications have potential to profoundly impact on performance and security. To reflect and quantitatively analyse such penalties, we propose a stochastic model to capture both the behaviour of virus attacks and the excessive load generated by the execution of anti-virus programs. As mentioned earlier, different modes of scanning plus other characteristics of such programs and attacks can partly be reflected in the relevant parameters of the model to ensure more accurate representation of real systems.

2.1.2 Detection and Response Approach

It is always wise to deter and prevent than cure. A computer network that is continually observed, assessed and evolved (e.g. by regularly applying security patches), can considerably shrink the attack surfaces [74, 103]. Regardless of what preventive measures are taken, a network can still become compromised given greater levels of motivations and skills [52]. Therefore, timely detection and notification of affected systems is crucially important. Nazareth and Choi [74] suggest that investing in security detection tools has a higher payoff than does deterrence investment.

Insiders vs Outsiders

The Computer Emergency Response Team (CERT)[1] Program's definition of an insider is an employee, a contractor or a business partner, either current or former, who, a) has or had authorised access to an organisation's network, systems or data; and b) exploits vulnerabilities to gain unauthorised access to such systems or data [94]. Due to growing reliance on technological infrastructures, organisations are made increasingly vulnerable to threats from insiders. Rich et al. [83] suggest that organisations would require both technical and behavioural controls to successfully deal with such threats. An outsider, on the other hand, is an individual or a group who seek to gain access to protected information from outside the organization, which usually involves using hacking techniques to break security keys and passwords or just take advantage of potential vulnerabilities in such networks.

Despite significant advances in prevention and detection techniques and algorithms, insiders are still to impose serious challenges on computer networks. There is no simple way of detecting insider threats before the attacks begin [83]. Moreover, there are too many known vulnerabilities for insiders to exploit; this is true due to their easy access and familiarity with such targeted systems [83]. In this research study, our proposed models assume smaller inter-vulnerability times for insider threats to reflect more realistic and shorter learning phases associated with these types of attacks.

Intrusion Detection and Response System (IDRS)

As computer networks are becoming ever more complex, there are always exploitable weaknesses such as design and programming errors as well as various socially engi-

[1] http://www.cert.org/

neered penetration techniques [114]. It is, therefore, widely accepted that prevention mechanisms, e.g. firewalls and encryption/decryption as previously discussed, alone are no longer sufficient to provide the desired protection for threatened networks. It is essential to also consider IDRS as a complementary measure to ensure improved security in computer networks.

IDRS involves capturing and auditing traffic and reasoning about it. The primary assumptions about IDRS include a) users and program activities are observable, say through network auditing mechanisms, and more importantly b) normal and intrusion activities have distinct behaviour [114]. IDRS can, therefore, be used to detect a wider range of attacks including network mapping (emanating, for example, from nmap), port scans, TCP stack scans, DoS bandwidth-flooding attacks, worms and viruses, OS vulnerability attacks, and application vulnerability attacks [50]. IDRSs are designed to help with early detection of malicious activities and properly respond to remedy or minimise the potential damage [24, 60, 114].

A taxonomy of IDRSs is introduced by Stakhanova et al. [97] where responses are, at the top level, classified by degree of automation and activity of triggered response. The automatic responses are further classified by the ability to adjust, time of response, cooperation ability or response selection methods. The authors also propose a set of essential features that an ideal intrusion response system should possess. A similar taxonomy can be found in [30].

There are two types of IDRSs, namely HIDS and NIDS. While HIDSs are pre-installed on host machines performing locally to determine if the node has been compromised, NIDSs are installed in the network level observing the traffic flowing in/out of the network. Like with anti-viruses, the effectiveness and performance of both HIDS and NIDS depends on the techniques and algorithms employed, e.g. misuse detection or anomaly detection, measured by their two inherent characteristics: the FN and FP probabilities/rates.

Bowen et al. [13] investigate the necessity of combined approaches in tackling insider threats. The authors believe that standalone detection mechanisms are often easily identified and avoided; therefore, a new design is proposed combining three complementary techniques, namely host-based user-event monitoring sensors, trap-based decoys and remote network-based detectors, to act synergistically with the goal of making it difficult for an adversary to avoid detection. The proposed prototype is shown to have meaningfully improved detection rate and accuracy. We consider a combined mechanism where the performance implications of the detection mechanisms is taken into account. In addition, our model provides different responses depending on the system security state. Moreover, we explicitly model FA so as to reflect their costly impact on performance and security of the real life systems.

In the event of a security incident, there must be an appropriate response process to tackle the problem effectively and in a timely fashion; this would, however, well depend on the types of application areas as well as the level of knowledge acquired from prior phases with regard to the incidents. For instance, [22, 24, 25] investigate secure group communication systems in Mobile Ad Hoc Network (MANET) where the emphasis is placed on military applications. More specifically, the authors consider types of network that are irrecoverable, implying that the whole system would fail and cease the operation once a security failure occurred. Goševa-Popstojanova et al. [35] and Madan et al. [60, 61], however, take a more generic and flexible approach to tackle security intrusions. They introduce a security-based model which allows the system to tolerate intrusions and continue its operation albeit possibly in a degraded mode. Their model presents a number of security states that a typical system may transition from one to another. Some of the states include Good, Vulnerable, Attacked, Triage, Fail Secure and Failed. While we further discuss their models in the following sections and chapters, it is worth mentioning here that their work does not consider the performance implications of security measures.

2.2 Security Modelling and Quantification

Adding more functionality to a system in the interest of security clearly requires more execution time. However, it is not easy to quantify the benefits achieved from any additional overhead. It is, therefore, very hard for performance engineers to argue whether or not a particular performance target should take precedence over a security goal [117, 118].

Littlewood et al. [56] and Brocklehurst et al. [14] intend to work towards operational measures for computer security. They believe a measure of security should quantitatively capture the intuitive notion of "the ability of the system to resist attacks", rather than how extensively safeguards have been introduced to a system during its design and development.

They propose that quantified security measures could involve expressions such as "the rate of occurrence of security breaches", similar to "the rate of occurrence of failures in reliability"; or "the probability that a specified mission can be accomplished without a security breach", by analogy with "the reliability function".

Authors of both papers, however, appear tentative in their conclusions due to some subtle differences between security and reliability environments. In particular, they are concerned with a) the presence of more variability in the threatening environments of security since security threats may be created randomly as well as deliberately, and b) the fact that whether the exponential distribution for 'the time to the next security breach' could be a justifiable assumption. As such, their reasoning by analogy approach is subjected to empirical investigation of several open questions identified.

Verendel [108] expresses similar concerns about a large part of research on security quantification methods. In his quite thorough survey of existing work, Verendel questions validity of the assumptions underlying the attempts at quantification of operational security. He argues that 'quantified security is a weak hypothesis' due to lack of validation and comparison between such methods against empirical data.

The main cause of this issue, according to the paper, is the lack of security data to validate the methods. While some risk factors associated with unreliable methods are emphasised, the paper makes suggestions to improve the knowledge about quantitative security.

Nevertheless, simulation and theoretical modelling are acknowledged as two broad approaches that can provide valid results with certain confidence levels providing that the underlying assumptions are well-supported by empirical means. In other words, it is clearly concluded that metrics and models need validation regardless.

It is well worth clarifying though that Verendel's critical analysis of security quantification mostly involves security models represented using analytic methods, e.g. sets of mathematical equations. As such, some of the concerns raised in the paper, such as randomness of security events, may be valid and require further investigation. They might otherwise be of less concern if alternative approaches, e.g. discrete event simulation (DES) techniques with more relaxed assumptions, are adopted. If in a model, for instance, some sojourn times required non-exponential distribution functions, using a simulation approach we could easily apply other best fit distributions readily available, including customised functions; this is not quite the case with analytic methods which strictly hold only for certain types under specific assumptions.

Jonsson and Olovsson [42] used empirical data collected from intrusion experiments to study a typical attacker behaviour. A hypothesis-based outcome suggests that an attacking process can be split into three phases: *learning phase*, *standard attack phase*, and *innovative attack phase*. While the probability for successful attacks during the *learning phase* and *innovative attack phase* is expected to be quite small - although for different reasons, this is expected to be considerably higher during the *standard attack phase*. Moreover, the insights from the data demonstrate that the breaches during the *standard phase* are statistically equivalent and that the time intervals between consecutive breaches are exponentially distributed, implying that traditional methods for reliability modelling could be applicable.

Security has for long been assessed from a qualitative point of view, but this needs to change [36, 60, 61, 109], and like reliability, security should be quantified in order to facilitate respective decision-making processes. The authors argue that it is no longer sufficient to prevent and/or even tolerate security intrusions; it is equally important to treat security as a QoS attribute at par with, if not more important than, other QoS attributes such as availability and performance.

Madan et al. [61], in particular, investigate various issues related to quantifying the security attributes of the SITAR[2], an intrusion tolerant system. Here a security intrusion (the attacker) and the response of an intrusion tolerant system to the attack are both modelled as random processes. Stochastic modelling techniques are employed to capture the attacker behaviour as well as the respective response of the system.

The security quantification analysis is first carried out for steady-state behaviour leading to measures like steady-state availability. By transforming this model to a model with absorbing states, similar to the models introduced in [18], they analyse transient behaviour of the system so that security metrics such as MTTSF and/or probabilities of security failure due to violations of different security attributes can be obtained.

While the importance of parametrisation and accurate adoption of parameters is emphasised, the focus is on methodology. Nevertheless, in the absence of exact values for the model parameters, a sensitivity analysis is conducted to help identify how (rapidly) the model behaviour can change with respect to various parameters.

Almasizadeh and Azgomi [5] propose a stochastic model for an attack process in order to obtain quantitative security metrics representing the security level of a system. The focus of their work is on how to model the progression of an attack process over time. They present an abstract state-transition diagram and transform it into a state-based stochastic model by assigning time distributions to its transitions. By

[2]Scalable Intrusion-Tolerant Architecture for Distributed Services

doing so, the attacker's and/or the defender's activities are implicitly considered in the transitions of the model. In other words, different distributions can accommodate for different behaviours of the attackers and/or the defenders (security mechanisms). The proposed state-based stochastic model has the Markov property where non-exponential probability distributions can possibly be involved. As a result, the model is considered as a Semi-Markov Chain (SMC) which is fully characterized if and only if the distributions of its transitions are known and characterized.

2.3 Performance vs Security Trade-offs

Recall that different systems are usually designed with different performance and security requirements in mind. We also realise that security mechanisms are likely to introduce undue computational and network overheads that may prevent performance goals from being met. As a result, trade-off analysis should be considered to help dynamically and mutually adapt performance and security as appropriate throughout a system life cycle.

Depending on the context and application requirements, either security or performance might be of more concern in which cases we should trade one off for the other. However, more often than not, we would like to achieve the best (optimum) of both performance and security combined. A model-based quantitative analysis, in particular, can provide system designers/administrators with a set of flexible tools enabling them to configure system parameters such that certain requirements are (will be) fulfilled. Such solutions may not necessarily be optimum for performance nor for security, but for their combination.

Cho et al. [16, 18–25] propose and evaluate a class of QoS-aware protocols for Secure Group Communication System (SGCS) in MANETs. The whole study is heavily dominated by model-based quantitative analysis in order to examine the trade-offs between security and performance and identify optimal design settings under which application-specific QoS requirements can be best satisfied.

Due to a mission-oriented design in mind, a distinctive characteristic of these systems and models is that they are considered irrecoverable, once failed. As a result, the lifetime of such systems is determined by a security-induced failure. They use an absorbing state[3] to explicitly model and signify the occurrence of security failure. Moreover, MTTSF, the average time to enter the absorbing state, is one of the main security metrics used to quantitatively reflect the level of security for the mobile groups under consideration.

They introduce a number of performance and security related input parameters which are shown, through trade-off analysis, having mutual impacts on one another. For instance, periodic batch rekeying is proposed as an efficient strategy to reduce the overhead by trading secrecy violation off rekeying overheads. It is also shown that an optimal batch rekey interval exists which minimises the cost per join/leave operation while satisfying the constraints in terms of delay and secrecy violation. Unlike the common belief that IDS should be executed as often as possible to cope with insider attacks and prolong the system lifetime, it is shown that IDS should actually be executed at an optimal rate in order to maximise the system lifetime.

We next review the model presented by Wang et al. [110]. Having employed a queueing model, the authors study performance and security of an E-Mail system composed of the user and the incoming mailbox together with some additional filtering mechanisms. There exist three measures obtainable, the *queue length*, a pure performance measure; the *information leakage probability*, a pure security measure; and the *mail system availability*, also a security measure since the Denial of Service (DoS) attack being considered here aims to disrupt the email service. The *information leakage probability* gives the steady-state probability of a mail being affected by an information gathering attack.

The system is modelled as though there existed four queueing systems each with its own distinct server - one serving regular emails and the others serving respective security related tasks. Such a model assumes no overhead on the email server im-

[3]A state, once entered, cannot be left.

posed by security attacks and/or their countermeasures, which renders the model less realistic. As a result, it appears that Wang et al.'s model in fact cannot be used for performance and security trade-offs analysis since performance and security are modelled as two independent aspects without mutually affecting each other. Nevertheless, the model may serve to explore the efficacy of different security-enhancing approaches.

Wolter and Reinecke [112] provide, in a great detail, a review of existing approaches in evaluation of performance and security. As acknowledged, there are no general models established to study performance vs security trade-offs. Most existing approaches instead take either security or performance as given and investigate the respective other. As a result, a simple yet reasonably generic model together with some simulation results are provided to illustratively evaluate the effectiveness of combined models.

The proposed model has two separate, possibly interlinked, submodels one of which captures the performance aspects of the system and the other takes into consideration the security related aspects. The approach allows for partially capturing the interrelationships between performance and security. In addition, it helps formulate some combined metrics between them. It is worth noting, however, that such a combined model assumes that:

- The general idea of performability analysis can be applied to the joint evaluation of performance and security. In performability modelling, one combines a performance model and a dependability model of the system with the goal of jointly evaluating performance and dependability, and the dependencies between parameters controlling either [68, 69]. Wolter and Reinecke, therefore, have taken a similar approach where the security model replaces the dependability model in a performability analysis.

- An encryption algorithm with longer security keys can provide better security [51].

In other words, Wolter and Reinecke rely on the analogy between security and reliability and make attempts to formulate optimisation problems for the trade-offs between security and performance as they exist between dependability and performance. It is, for instance, argued that software rejuvenation is used to increase dependability; this, however, comes at a cost. System failures incur a cost too. The optimisation problem then is to tune the rejuvenation interval in such a way that the total cost, here the sum of the rejuvenation cost and the downtime cost, is minimised.

Analogously, they formulate an optimisation problem for encryption. To improve security, an encryption mechanism can be employed, which would come at a cost (of further delays). At the same time, a security incident (system failure) has associated costs due to increased downtime and/or probability of data breaches. As a result, there should be an optimal encryption key length for which the total cost - of both the encryption process and the security incident together - can be *minimised*. They also consider the revenue achievable by operating the system. Here, the encryption cost and, at a much higher degree, the time for recovery from a security incident would result in a reduced revenue. In order to *maximise* the revenue, however, the encryption key must be chosen such that the encryption cost is reasonably low and the security incidents only occur very rarely.

Below is, for ease of access, the proposed model [112] redrawn in Figure 2.1. Using the Generalised Stochastic Petri Net (GSPN) paradigm, the model represents a simplified communication system where each newly generated packet first needs to be encrypted in order to be eligible for transmission. From security perspective, the system can only be, at any given time, in either the *Secure*, *Insecure* or *Restoring* state. An inhibitor arc freezes the encryption process while the system is recovering from a security attack. Moreover, an abstract combined performance and security measure together with cost and revenue metrics are formulated, which explicitly express the trade-offs between security and performance. It is also shown that system parameters can be found that optimise those combined metrics; while such parameters are not optimal for performance nor for security, but for the combination of both.

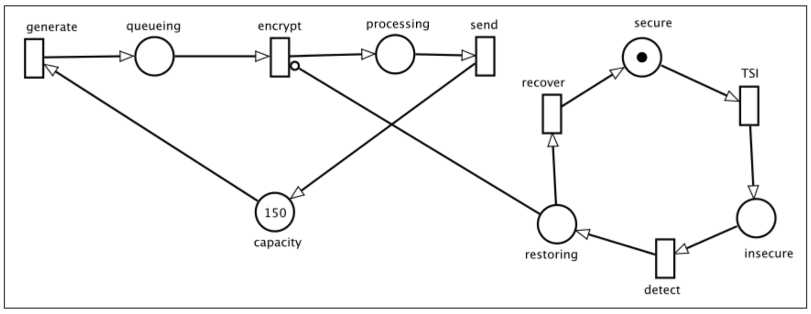

Figure 2.1: Combined performance and security model [112], using the GSPN paradigm to represent a simplified communication system with an encryption process. The performance submodel is connected to and affected by the security submodel using an inhibitor arc. In the event of a security attack, once detected, the inhibitor arc freezes the encryption process preventing it from any further operation until the system is fully recovered.

A research work with similar approach has recently been published by Meng et al. [67] and Meng [66] who investigate the trade-offs between performance and security in mobile offloading systems. While migration of complex computations from mobile devices to more powerful servers on the cloud may significantly improve performance and energy consumptions, there are security implications associated with increased data transmissions over the network with potentially unknown threats.

Meng et al. [67] introduce a hybrid Continuous Time Markov Chain (CTMC) and queueing model which takes into account the behaviour of both the system and the attacker. Both steady-state and transient analyses of the CTMC model are carried out, in analogy to dependability, which enables the quantitative assessment of performance and security attributes of the mobile cloud offloading system subject to timing attacks. Similar to previous studies, the MTTSF metric is used as the measure for quantifying the security of the offloading system. For the transient analysis the compromised state of the CTMC model is transformed into an absorbing state, as expected.

Under a timing attack scenario, the attacker continually sends jobs to the server and measures the run times in an attempt to help guess the security keys. To improve security, the server would need to rekey, from time to time, to generate new pairs of keys (public/private) for encryption/decryption of the messages exchanged between the client and the server. However, Rekeying comes at a cost; at the same time, security breaches due to compromised keys cost too. As such, quantitative modelling of performance and security is employed so as to help analyse and identify optimal interval rekeying on the server to ensure high security and low performance cost.

Montecchi et al. [72] study the trade-offs between scalability and security on the OPENNESS[4], a web-based platform providing different kinds of services to different groups of users. Since scalability is usually intended as a metric that links the size of a system with its achievable performance [37, 58], the evaluation here, instead, focuses on the performance metrics and their sensitivity with respect to the size of the system. A stochastic modelling approach is adopted for quantitative analysis of bottlenecks as well as performance implications of security countermeasures in the target system.

As an experimental case study, a relatively large fictitious OPENNESS platform with a reasonable number of services, actions, users and user profiles is designed. The modelling activity is then carried out using the Stochastic Activity Network (SAN) formalism, making full use of its characteristics of modularity, reusability and maintainability. The analysis model is realized through the composition of a set of predefined template models, which facilitates the construction of the overall system model, and the evaluation of different configurations by composing them in different ways. The methodology for design, implementation and parameterisation of the model is presented in the paper in great detail.

The paper begins defining a reference scenario with default parameters, which is meant to serve as the basis of comparisons. In the second scenario, they authors aim at evaluating scalability of the system with respect to the number of users although this can be done against any other system parameters too. Third scenario employs two security countermeasures that are believed to have significant impacts on system performance. *Utilisation* and *Mean waiting time* are used to evaluate the steady-state performance of the OPENNESS platform across all scenarios above.

[4]OPEN Networked Enterprise Social Software suite

The simulation scenarios together with the experimental results clearly demonstrate the usefulness of this quantitative model when it comes evaluating such large enterprise platforms utilising a similar approach as ours. This can be particularly helpful as it allows for scaling up any system parameter of interest so that one can analyse and identify potential bottlenecks in the system ("*what-if*" analysis).

On the other hand, the model lacks the mutual interaction between performance and security. As we have seen earlier in this thesis, not only higher security comes at a cost, but very low security also can cause serious performance and security issues. In Montecchi et al.'s model, the impact of security measures is *only* rendered as further delays in service times. It does not, for instance, take into consideration the possibility of data breaches and/or increased downtimes due to lower security levels potentially leading to frequent security failures. As a result, it appears that the proposed model, as is, cannot be used to carry out the analysis of optimal trade-offs between scalability (performance) and security.

Chapter 3

Formalisation of Security

In this chapter, without loss of generality, we remove our attention from a computer network down to an individual computer node[1], for which we aim to conceptualise and formulate security so that it can be modelled and quantified. Measuring security of an individual entity can then lead to measuring security of a computer network, which simply consists of multiple entities [78]. We discuss security of a computer network further in our future work.

Given that the node is subject to security threats, in Section 3.1, we briefly review the existing work with respect to security formulation and modelling. In Section 3.2, we introduce a state transition diagram which represents distinct security states, where the node can be found at any given time, together with possible transitions between them. In Section 3.3, we conceptualise and formulate typical behaviours of security threats and their respective countermeasures. And Section 3.4 provides brief definitions for some standardised security metrics that are used throughout this thesis to measure security quantitatively in the context of an individual node.

3.1 Existing Models

The study of existing work reveals that the idea of investigating security in a similar way as dependability has been around for a long time [9, 14, 18, 36, 41–43, 56, 61, 75, 106, 109, 119]; however, pragmatic studies are rare [112].

[1]In this thesis, we use interchangeably "computer entity", "computer node", "computer system", "entity", "host", "node", "computer" or "system" to refer to a single computer entity.

Goševa-Popstojanova et al. [35], Madan et al. [61] and Griffin et al. [36] collaboratively propose a generic security model that enables multiple mitigation strategies to exist and supports mitigation of intrusions with different impacts. We note that their model can in fact describe a class of different systems with varying features. For instance, the model represents a system which is a) capable of, to some extent, tolerating intrusions and b) degradable and can maintain to meet the service requirements (possibly limited services) even when the environment is hostile. If a system lacked any of these presumed features, the model by all means would have to be updated, e.g. by removing the respective states and transitions, so as to properly reflect the missing functionality.

Their proposed state transition diagram in its general form, as depicted in Figure 3.1, encompasses the following list of states:

- G: the good (secure) state of the systems (by default in the beginning)
- V: the vulnerable state; during the vulnerability identification phases of an attack
- A: the active attack state; damage may follow
- TR: the triage state; once in here, the system may be able to recover or limit the (possible) damage (depending on the type of the attack)
- GD: the graceful degradation state where only essential services are maintained
- FS: the fail-secure state; stops functioning (depending on the type of attack)
- F: the failed state; if all existing strategies fail; once in here, an alarm is signalled
- UC: the undetected compromised state
- MC: the masked compromised state

We note that the system is returned to state G from UC, F, FS or GD by going through a restoration/reconfiguration/evolution process, implying that an appropriate fix for the exploit is applied. [36].

While the proposed security model is reasonably generic and can be flexibly adapted to represent a large collection of real-life systems, it is viewed purely from security perspective. In other words, whether/how each of these security states is going to have any implications for performance appears beyond the scope of these studies.

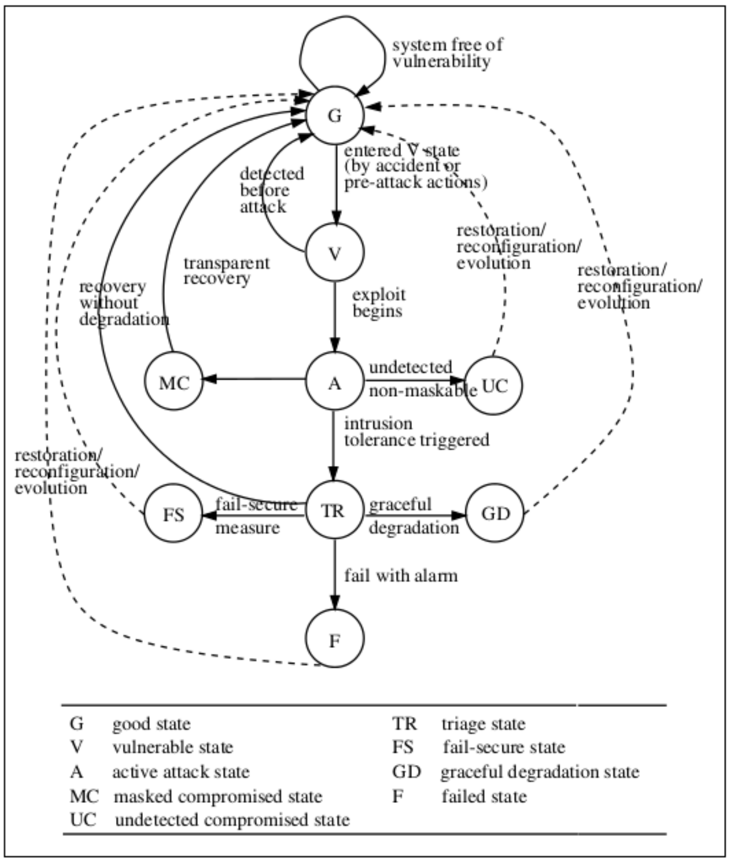
system free of vulnerability
G
entered V state (by accident or pre-attack actions)
detected before attack
V
transparent recovery
recovery without degradation
restoration/ reconfiguration/ evolution
restoration/ reconfiguration/ evolution
exploit begins
MC
A
undetected non-maskable
UC
intrusion tolerance triggered
restoration/ reconfiguration/ evolution
FS
fail-secure measure
TR
graceful degradation
GD
fail with alarm
F
G good state
V vulnerable state
A active attack state
MC masked compromised state
UC undetected compromised state
TR triage state
FS fail-secure state
GD graceful degradation state
F failed state

Figure 3.1: A generic state transition diagram for a class of different intrusion tolerant systems [61]. The model represents systems with multiple mitigaton strategies in place. Due to its high flexibility, it can be easily trimmed to represent systems of varying features.

Recall that Cho et al. [19–21] also take a quantitative model-based approach to study trade-offs between performance and security. While Cho et al. do not explicitly propose any state transition diagram for security, they do introduce some similar distinct security states. As shown in Figure 3.2, for instance, the T_m, UC_m, DC_m and GF places, in Stochastic Petri Net (SPN) formalism, play similar roles as the G, UC, (A and/or TR) and F states in Goševa-Popstojanova et al.'s more generic state transition model, respectively.

It is worth mentioning, however, that Cho et al.'s models represent a specific class of systems and applications, in particular SGCS in MANET with mission-critical applications. As such, there are differences in features and functionality too, as expected; for instance, here

- the system is not repairable; once failed, it cannot be recovered. As a result, the GF state is an absorbing state instead.

- The entities moving through states appear to be individual computer nodes (members of the SGCS). This, however, suddenly changes in case of the GF state, where the entity is now the whole group.

- While FNs are implicitly modelled, there is an explicit link to accommodate FPs.

- Once in the DC_m state, the recovery (eviction of compromised nodes) may be delayed in an attempt to trade security off for better performance.

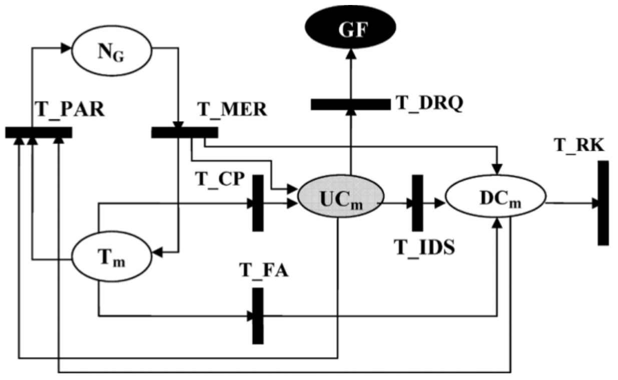

Figure 3.2: A Stochastic Petri Net (SPN) model which captures behaviours of a mission-critical SGCS, instrumented with an IDS to deal with the insider attacks in MANET [18]. The systems being modelled here are assumed irrepairable; that is, once in GF, for whatever reason, there is no way out and the whole system comes to a halt.

Yet another model for quantification of security is proposed by Wolter and Reinecke [112]. Analogous to dependability, where a system is assumed to be either *working* or *failed*, Wolter and Reinecke describe the security state of a system being in either *secure*, *insecure* or *recovery.* As drawn in Figure 2.1, the *secure* state represents the normal operating state of the system. The *insecure* state is reached when a security incident occurs, which is by analogy with a *failure* in the world of dependability. And the *recovery* state reflects the state of failure, where actions are required in order to have the system cleaned and recovered from security attacks. Once recovered, the system is considered secure and can once again start operating as normal. The authors specify the transitions between the states either by probabilities, continuous random variables, or stochastic processes.

Wolter and Reinecke's model is composed of two submodels: a) the security state transition submodel and b) the packet transmission submodel. The security submodel can be seen as a simplified case of Goševa-Popstojanova et al.'s model representing a system which, from security perspective, lacks some of the features discussed earlier. For instance, the system here is incapable to identify vulnerabilities; nor does it possibly own any degradable modes. Instead, the model here introduces a very useful link to interconnect the security and performance submodels so that performance implications of security failures can be partially captured. An inhibitor arc is used to freeze the encryption functionality when the system is in the *recovery* state.

Vulnerability Window

Vulnerabilities appear to be playing an important role in the security state transition cycle of any system. Below we briefly review the concept of *Vulnerability Window* before we delve into the details of our contribution in this chapter.

Arbaugh et al. [6] propose a life-cycle model with distinct states that a vulnerability can enter during its lifetime. More specifically, a vulnerability may transition through the following states:

- **Birth** denotes the flaw's creation; it usually occurs unintentionally during development. However, if the birth is malicious and thus intentional, then birth and discovery coincide.
- **Discovery** indicates the moment when someone discovers security or survivability implications in a product; it is only then the flaw becomes a vulnerability.
- **Disclosure** takes place when the discoverer reveals details of a vulnerability to a wider audience.

- **Correction** takes place if a vulnerability is correctable and the vendor or developer releases a software modification or configuration change that fixes the underlying flaw.

- **Publicity** is when a vulnerability becomes publicly known via various news channels.

- **Scripting** denotes the period when a cracker scripts the exploitation so that anyone with little skills can compromise systems by exploiting the vulnerability.

- **Death** takes place when a vulnerability dies; for instance, when the number of systems it can exploit shrinks to insignificance.

We note that all these states are causally related and that they must always occur in order, in particular the first three states - **Birth**, **Discovery** and **Disclosure**. Figure 3.3 shows how the number of intrusions should be increasing once users discover a vulnerability. Note that the rate continues to increase until the system administrators release a patch or workaround. We realise that the life-cycle model together with the detailed information about the intrusion rates (Figure 3.3) can inform the respective system parameters such that the system can be, with respect to vulnerability, configured and parametrised more realistically.

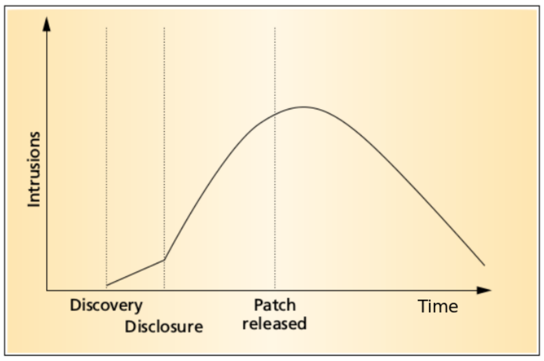

Figure 3.3: Intuitive life cycle of a system-security vulnerability [6], illustrating how a vulnerability may transition through a number of states from Birth to Discovery, Disclousure, Correction, Publicity, Scripting and Death.

Arbaugh et al. also suggest that security of an information system can be envisaged to transition between several distinct states, namely *hardened*, *vulnerable*, and *compromised*, during its lifetime. The model appears quite similar to what we have explored thus far with regard to the security state transition model. Here, a system can be considered

- *hardened* if all vulnerability patches have been installed
- *vulnerable* when there is at least one security-related correction that has not yet been applied
- *compromised* if one or more vulnerabilities have been successfully exploited

A system typically oscillates between *hardened* and *vulnerable* states (Figure 3.4). As a system design factor, one should make efforts to reduce the time the system spends in the vulnerable and compromised states [6].

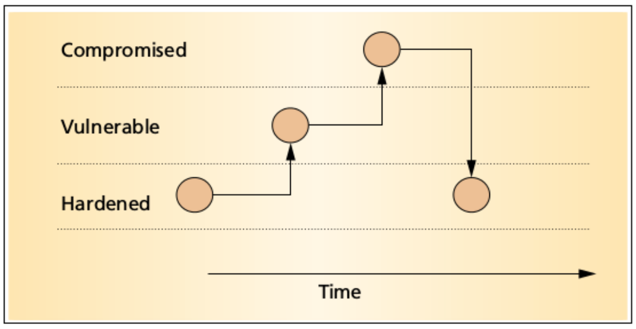

Figure 3.4: Host life cycle [6], showing how security of a typical system moves from its secure (Hardened) state through a vulnerable state and then becomes insecure (Compromised). The horizontal timeline illustrates the time a system spends in the vulnerability and/or compromised states and how the security can possibly be improved.

3.2 Security State Transition Diagram (SSTD)

In this section, we aim at formulating the concept of security for a single computer entity, which is part of and connected to a generic computer network. Let us assume that the node is

- subject to three different types of security threats, namely viruses, insiders and outsiders.
- well-equipped with three security countermeasures of anti-virus programs, IDS and encryption mechanisms, respectively.

- repairable; that is, it is capable of being restored to a fully operational condition [73], provided a (security) failure is detected by, at least, one of its security countermeasures.
- not degradable nor are there any redundant resources available; that is, the node is unable to tolerate security intrusions, neither fully nor partially (where system services could be flexibly downscaled).

We propose a Security State Transition Diagram (SSTD) model on the basis of the complementary security quantification solutions that we have just reviewed above from the existing work. It inclusively accommodates for four tangible[2] states [62], together with all possible transitions between them. More specifically, as illustrated in Figure 3.5, the four distinct, mutually exclusive states that the computer node can possibly sojourn in include *Secure*, *Vulnerable*, *Insecure* and *Restoring* states.

Recall that this research thesis is mainly concerned with model-based analysis of trade-offs between performance and security, in which the performance implications of security and vice versa are of paramount importance. We therefore make slight amendments to the solution to ensure that we can in addition capture a) explicitly, the respective FA signals of the security measures in place, and b) fully, the implications of each security state for performance. Below comes a description of each of these security states in greater details as follows.

The *Secure* State is the default state indicating that the system is perfectly secure and that there are no vulnerabilities known, as yet. While in here, there are two possible trajectories for the system to move to. It can transition to either the *Vulnerable* state, in the event of any vulnerability discovered (by an attacker), or the *Restoring* state, in the event of a FP signal issued (by a security measure); the imminent event dictates which one of the two possible transitions to occur. In the absence of these security events, however, the self-transition (depicted as a directed

[2]Petri Net (PN) states (markings) in which no immediate transitions are enabled are called tangible as opposed to vanishing states in which there is at least one immediate transition enabled. Moreover, the system spends a positive amount of time in tangible states, and a null time in vanishing states.

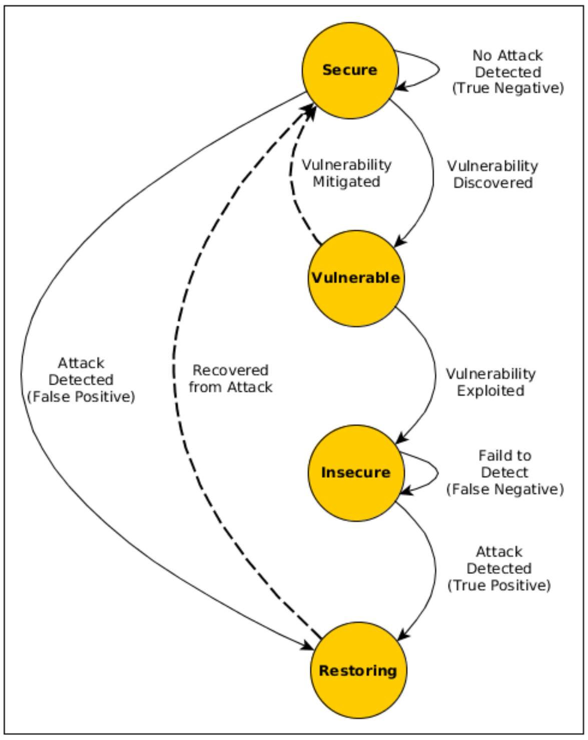
Secure
No Attack Detected (True Negative)
Vulnerability Mitigated
Vulnerability Discovered
Vulnerable
Attack Detected (False Positive)
Recovered from Attack
Vulnerability Exploited
Insecure
Faild to Detect (False Negative)
Attack Detected (True Positive)
Restoring

Figure 3.5: Our proposed security state transition diagram model, showing four tangible states and all possible transitions between them. The four distinct, mutually exclusive states include *Secure*, *Vulnerable*, *Incecure* and *Restoring* states.

loop) indicates no state changing, implying that the system will continue to stay in the *Secure* state.

The longer the sojourn times in the *Secure* state, the better. To improve the overall sojourn time in this state, system designers can consider: a) employing more advanced and precise security measures (with lower FP rates) and/or b) minimising vulnerability patch times, which can improve the likelihood of the system being taken back to the *Secure* state after a vulnerability is discovered.

It is worth reminding that the system incurs extra performance penalties (increased unavailability) due to FP alarms; that is, despite the system actually being secure, it is falsely taken to the *Restoring* state where it will have to cease to operate all its normal functionality and carry out an unrequired recovery! Therefore, the lower the FP rates, the better.

The *Vulnerable* State indicates that the system has some known (discovered) vulnerabilities and that it is susceptible to attacks although no security attack has begun to exploit them as yet. While in here, there is a possibility that the system is once again taken back to the *Secure* state provided those vulnerabilities are adequately quickly patched, prior to vulnerability exploitation. Otherwise, an active attack will take place, leading the transition to the *Insecure* state, Compromised Undetected (CU). It is important to realise that the longer the sojourn times in this state, the higher the chance of exploitation [7].

The *Insecure* State (Compromised Detected (CD) indicates that an active security attack has taken place without the system administrators' knowledge. In other words, the system has been compromised but the administrator team is unaware of! While in here, the system has no way out unless a security mechanism detects the actual attack (True Positive (TP)) in which case the system is taken to the *Restoring* state. The *Insecure* state is perhaps the most worrying and costly period of a security cycle due to the fact that the intruders have successfully gained access to system resources without even being detected!

Despite the presence of security measures, it is possible that they will fail to detect the intrusion (FN). For instance, given a security measure with FN $= 0.05$, the probability that it would fail to detect an actual attack in its first inspection or in its first and second inspections would be 5% and $(0.05 * 0.05) = 0.25\%$, respectively. As a result, the higher the FN rates, (possibly) the longer the sojourn times in this state, and the higher the probability of data breaches (more damage) [7]!

The *Restoring* State is reached when a security mechanism detects an intrusion, truly (TP) or falsely (FP). While in here, a full recovery is enforced under the system admin control. It is wise to expect that all normal functionality of the system temporarily will cease to operate until it is fully recovered from the security failure and taken back to the *Secure* state for another fresh cycle. This is in accordance with the assumptions made for the node, in particular lack of redundancy or degradation mode. It is evident that recovery comes at a cost and may significantly contribute to the overall downtime of the system. The model can therefore inform system designers to work towards minimising these sojourn times by a) accelerating the recovery processes, and b) ensuring that the system rarely ends up in this state in the first place. Further analysis can help identify which system parameters are most contributing to the transitions leading to this state.

Proliferation of the SSTD Models

We note that each security defence tool is designed to effectively target a certain class of security threats [82]. As a result, we propose three SSTD models, one for each class of security threats together with its respective countermeasure as assumed above; more specifically,

- The Virus Security State Transition Diagram (VSSTD) captures the behaviours of attacks originated by viruses and those of the anti-virus programs.
- The Insiders Security State Transition Diagram (ISSTD) captures the behaviours of attacks originated by insiders and those of the IDS.
- The Outsiders Security State Transition Diagram (OSSTD) captures the behaviours of attacks originated by outsiders and those of the encryption mechanisms as well as the NIDSs.

While all three SSTD models are very much similar in the concept and design, they work independently from one another. The overall security state of the system, as an integrated entity, is then decided based on the state information collectively gathered from each of these individual SSTD models.

Below are different cases demonstrating how the state of security for the system as a whole could be concluded based on the states of the individual SSTD models as follows:

- The overall node is considered *Secure* if and only if all three SSTD models are jointly in their *Secure* states.
- If at least one SSTD model is in its *Vulnerable* state and the rest are in their *Secure* states, the node is considered *Vulnerable* overall.

- If at least one SSTD model is in its *Insecure* state, the node is considered *Insecure* overall, regardless.
- If at least one SSTD model indicates the system being taken to the *Restoring* state, the node is considered in *Restoring* frozen, overall.

As expected, the methodology appears to remain the same and the number of SSTD models can flexibly scale up/down depending on whether one plans to take a holistic approach or just to include a minimum number of countermeasures to battle against the most common security threats.

It is crucially important to realise that security is usually seen as a chain and, as such, it is only as strong as its weakest link [15]. In other words, although employment of longer security keys and stronger encryption algorithms is likely to improve the security from the perspective of the outsiders, it may not necessarily improve the overall security of the system! To do so, we need to effectively improve the security from all perspectives where every aspect of security threat and the respective countermeasures should be considered.

It is worth mentioning that we have implicitly assumed that multiple attacks of a certain type are not allowed; that is, there can only exist one attacker, at most, of each type at any given time. If an insider, for instance, has just exploited a vulnerability and compromised the system (already in the *Insecure* state), the model will not allow for any more attacks to be originated by insiders in parallel (not even in learning phases), unless the one currently circulating through the ISSTD is either mitigated or eventually caught by the IDS installed in the node.

Further Research

- We realise that, in a real-life, there is no way to limit the number of attackers simultaneously targeting a system. In other words, while it is sensible to allow for simultaneous attacks across different attack types, as in this research thesis from insiders, outsiders and viruses, it should also be perfectly fine to assume that there might be more than one for each of these attack types at any given moment who are making attempts to intrude the system.

- Recall that the SSTD models are assumed to work independently. However, it does not always make sense to assume a vulnerability is known to a certain group of attackers but not to the others. In a real-life, if a vulnerability is disclosed, one should expect all types of attackers to become soon after aware of it, hence a dependency among SSTD models!

3.3 Security Threats and Measures

In this section, we formalise security threats and security mechanisms. In particular, we look at the typical behaviours of security threats, attackers, in conjunction with the mitigation actions taken by security measures that are responsible for tackling such security threats.

We assume that the security measures are configured to execute security inspections on a periodic base and that both the inspection times and the time intervals between two consecutive inspections are random processes with probability functions *exponentially* distributed. We also note that the attackers and the security measures behave and act quite differently with respect to the current phase of attack (the security state).

Below comes, in details, an explanation of a) how an attack is initiated; b) how it possibly makes its way to our individual computer node; and c) how security measures on the node deal with the attack and bring the system back to normal again. Due to so much similarity in the SSTD models, we rather use more generic terms of "attacker" and "security measure" related to the SSTD model illustrated in Figure 3.5. Nevertheless, the same applies to all three SSTD models with certain classes of attackers and their countermeasures as discussed earlier in this chapter.

Recall that our individual computer entity is initially assumed secure with no known vulnerabilities, implying that the SSTD model is in its *Secure* state. At this stage, the attacker makes a fresh start spending some time to learn about the system and its potential security holes. In this research thesis, without loss of generality, the learning time for the attacker is assumed to follow an exponentially distributed function although any arbitrary stochastic function of interest can be adopted.

While in the *Secure* state, two independent, mutually exclusive random events are possible to take place to change the state of security; that is, they cannot happen together, and the occurrence of either prevents the other from happening. More specifically, if a vulnerability is first discovered, the attacker initiates the change and moves the security state to the *Vulnerable* state, implying that the attacker with some skills and gained knowledge has now the potential to attack the system. Alternatively, if the security mechanism falsely signals a threat (FP), it causes the transition to the *Restoring* state. The latter partly contributes to the probability of the system being unavailable.

Once in the *Vulnerable* state, the main objective of the attacker is how to exploit the existing vulnerabilities and commit active attacks. At the same time, the system administrators work hard to remedy the known vulnerabilities at the shortest possible time and ensure that the system is once again in the *Secure* state. A competition is therefore launched between the attacker and the administrators and the next trajectory from here will be decided based on the winner of this competition. If the attacker wins, the system will transition to the *Insecure* state, implying that the system is compromised but undetected (CU); that is, the system and its administrators are not aware of such an incident yet despite the intruder having gained access to the resources! This is the most undesirable condition and can be costly due to a high likelihood of data breaches.

However, if the administrators are able to patch the vulnerabilities prior to any exploitation, the system can be taken to the *Secure* state once again, where the attacker will have to start from scratch! This is due to the memoryless property of the exponential distribution which implies the absence of ageing and learning [61]. Although this may not appear appropriate for modelling behaviours of an attacker, there are other complexities involved too which need to be considered together. For instance, while it makes sense that the obtained skills and knowledge are taken into account in the attackers' future attempts, it is also perfectly reasonable to assume that the system administrators have their own learning curves and are increasingly becoming more experienced and quick in their actions! It is, therefore, sensible to consider dynamic adoption of activity rates for both the attackers and the system administrators (as well as security measures). We discuss this in our future work and propose some extensions.

Suppose that the attacker has exploited the vulnerability, hence in the *Insecure* state. This implies that the system is already under an active attack and the intruder has gained access to the resources. From the attacker's point of view, the longer the sojourn time in this state, the better! Once in here, the system is left at the mercy of the attacker unless the security measure brings this to an end; once the attack is detected, the system transitions to the *Restoring* state. There is a possibility, however, that the security measure fails to detect the attack (FN) in which case, the system needs to experience longer stays at this state, potentially leading to even more damage.

During the *Restoring* state, the system ceases all its normal operations and undergoes a full recovery. It also makes sense to disable the line of the attack while in this state. Once recovery is completed, the system is once again taken back to the *Secure* state and all its normal operations together with a line of attack are activated. We note that, while in the *Restoring* state, the system is completely non-operational. To reduce performance implications and the associated cost, one should either reduce the sojourn times in the *Restoring* state or ensure that the events leading to that state become less frequent.

Of paramount importance is the implications an attacker and/or a security measure may have for performance of the system under consideration. At a minimum, this can include the extra computational power required by the security measure in order to periodically execute. This research thesis has currently taken into consideration performance implications of security measures as well as security failures. We discuss these in further details in the Chapter 4.

Further Research

In addition to security measures, some attack types may be computationally intensive too; the DDoS is probably the most common attack of this type, which is usually accomplished by flooding the system with superfluous requests in an attempt to overload it and prevent some or all legitimate requests from being fulfilled. In its less destructive form, an attacker may even attack the target system in order to solely take advantage of its resources without intending to cause any harm at all. Such performance implications are not considered in this research work. We do, however, plan to conduct further research in this regard in our future work.

3.4 Security Metrics

Security metrics have received significant attention. However, they have not been systematically explored based on the understanding of attack-defence interactions [78]. While the terminologies used by the community are varying, we use standardised consensus-based security metrics established by The Center for Internet Security (CIS) [28] for consistency. Below we discuss a selective list of those metrics which we base our discussions in the rest of the thesis on as follows.

- **Mean Time to Mitigate Vulnerabilities (MTTMV)**: The average time taken to mitigate known vulnerabilities. The less time required to mitigate a vulnerability, the more likely to react effectively to reduce the risk of exploitation of vulnerabilities.

- **Mean Cost to Mitigate Vulnerabilities (MCMV)**: The goal of this metric is to understand the effort required for vulnerability remediation activities.

- **Mean Time to Patch (MTTP)**: The average time taken to deploy a patch; the more quickly patches can be deployed, the lower the mean time to patch and the less time in the *Vulnerable* state.

- **Mean Cost to Patch (MCP)**: The goal of this metric is to understand the effort required for vulnerability remediation activities.

- **Mean Time Between Security Incidents (MTBSI)**: The average time, in days, between security incidents; analogous to common reliability metric of MTBF.

- **Mean Time to Incident Discovery (MTTID)**: The effectiveness in detecting security incidents; the faster in detecting an incident, the less damage it is likely to cause.

- **Mean Time to Incident Recovery (MTIR)**: The effectiveness in recovering from security incidents; the sooner to recover from a security incident, the less impact the incident will have.

- **Mean Incident Recovery Cost (MIRC)**: The cost of returning business systems to their pre-incident condition.

- **Number of Incident (NOI)**: The number of security incidents for a given time period.

- **Mean Cost of Incident (MCOI)**: The mean cost from security incidents identified relative to the number of incidents occurred during the metric time period.

Below are a few more security related parameters that we have used in this thesis; however, they do not appear to have been documented by CIS.

- **Mean Time To Virus Vulnerability (MTTVV)**: The average time that it takes for a virus to discover a vulnerability, once in the *VSecure* state.

- **Mean Time To Virus Attack (MTTVA)**: The average time that it takes for a virus to commit an active attack, once in the *VVulnerable* state.

- **Mean Time To Insider Vulnerability (MTTIV)**: The average time that it takes for an insider to discover a vulnerability, once in the *ISecure* state.

- **Mean Time To Insider Attack (MTTIA)**: The average time that it takes for an insider to commit an active attack, once in the *IVulnerable* state.
- **Mean Time to Outsider Vulnerability (MTTOV)**: The average time that it takes for an outsider to discover a vulnerability, once in the *OSecure* state. This is one of the key parameters since it is used to quantify security levels (determined by application of different security key lengths) and map them in terms of the time it takes for an outsider to discover a vulnerability; that is, the longer the security key length, the longer the time it takes to find a vulnerability.
- **Mean Time To Outsider Attack (MTTOA)**: The average time that it takes for an outsider to commit an active attack, once in the *OVulnerable* state.

For a comprehensive detailed list, interested readers are advised to refer to the original document [28].

Chapter 4

Nodal Modelling and Analysis

In this chapter, we work towards proposing a combined quantitative model which allows for capturing both performance and security aspects of the individual node more realistically. More specifically, Section 4.1 gives a brief introduction to the modelling formalism and solver employed in this research work. Section 4.2 discusses the nodal model including all its submodels in a great detail.

4.1 Modelling Formalism

A formalism is a language for expressing a model within a framework. Formalisms are many and varied including Queueing Network (QN), PN and Performance Evaluation Process Algebra (PEPA) [79], among others. More often than not, the original formalisms have limitations leading to less expressive, inefficient or even inaccurate models which are unable to capture some key characteristics of the systems under study. New formalisms, therefore, are continually developed from their predecessor(s) in an attempt to improve expressiveness of the models as well as facilitate modelling and evaluation of more sophisticated functionalities with desired levels of details. Here in this thesis, we use the Stochastic Activity Networks (SANs) formalism, which is one of many extensions of the well-known PN formalism.

Let us begin by providing a working definition of a typical model which, in the context of SAN formalism, is composed of *states* and *actions*. The *states* of the model represent an abstraction of the system's behaviour and characteristics with respect to the time. They can be divided into distinct, non-overlapping state variables the union of which will give the total states of the abstracted system. *Actions*, on the other hand, define the way the states of the model may change. When an action occurs, it induces the change of *states* associated with it [100].

4.1.1 Stochastic Activity Networks (SANs)

SANs have been used since the mid-1980s for performance, dependability, and performability evaluation [90]. They have been utilised as a modelling formalism in three modelling tools (METASAN, UltraSAN, and Möbius), to evaluate a wide range of systems.

SANs can distinguish between two types of modelling formalisms: *atomic* and *composed*. An *atomic* model is completely defined in one modelling formalism and is a self-contained representation of either part of a system or the entire system. On the other hand, a *composed* model is a collection of other models joined together by sharing *states* or *actions*. It should be noted that a *composed* model may be joined to other *atomic* or *composed* models to form a new *composed* model [100]. In this thesis, we use the SAN *atomic* formalism to design the independent modules and then the *composed* formalism to integrate them all into one single model representing the individual computer node.

4.1.2 Möbius: The Modelling Tool

Despite the development of many modelling formalisms such as PN, various extensions of QN and PEPA, and various model solution methods such as simulation, analytical solution and state space exploration, most tool implementations support only a single formalism and usually a single technique. Furthermore, due to lack of cross-compatibility, models expressed in one formalism cannot be combined with models expressed in other formalisms. This monolithic approach both limits the usefulness of such tools to practitioners, and hampers modelling research, since it is difficult to compare new and existing formalisms and solvers [29].

Möbius, however, uses a method to eliminate these limitations. In fact, it is developed based on the belief that no one modelling formalism can be the best way to build all models of systems from across the diverse spectrum of application domains [88]. Möbius supports multiple interacting formalisms and solvers by providing an infrastructure which is extensible and new formalisms and solvers can be added to the tool without changing those already implemented. This can be done through the use of an Abstract Functional Interface (AFI), which provides a formalism-independent interface to the models. It allows models expressed in multiple formalisms to interact with each other, and with multiple solvers [29].

The driving motivation behind the framework is that a specific modelling formalism, such as SAN or QN, may be appropriate for the representation of a particular portion of a system being modelled, but not appropriate for an entire system. If a user is allowed to specify each part of the system in an appropriate representation, it becomes easier for the user to attain accurate results using simulation and analytical numerical methods [100]. Furthermore, since models are constructed in specific formalisms, the expressive advantages of the particular formalisms are preserved [89].

Although Möbius was originally developed for studying the reliability, availability, and performance of computer and network systems, its use has expanded rapidly. It is now heavily utilised for quantitative modelling and evaluation of a broader range of discrete-event systems, from biochemical reactions within genes to the effects of malicious attackers on secure computer systems, in addition to the original applications.

4.2 Nodal Model

We now continue to work towards proposing a model-based solution for our individual node such that a) behaviours of both security attacks and security measures are quantitatively captured; b) performance implications of the security elements are identified and accurately represented; and c) any interrelations between performance and security are recognised and realised. Our quantitative model, as expected, builds on the SSTDs introduced in Chapter 3.

The Problem Scenario We assume that a simplified single server computer node is given that is solely responsible for transmission of packets[1] to the other nodes in the network.

In a perfectly secure environment, e.g. one of the extreme cases mentioned in Chapter 2, the node could be well represented with a simple SAN model of a single queue with finite capacity (Figure 5.3). Such a node would be expected to work towards and achieve its near nominal performance since there would be no security related tasks which could continually be disrupting the server.

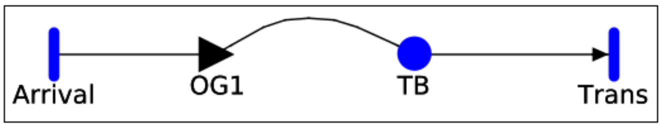

Figure 4.1: A single server finite-capacity queueing node represented using SAN Atomic Formalism. The node is assumed to be solely responsible for packet transmission and that there are no security threats whatsoever; as such, no security measures are employed either.

Unfortunately, there is no such an environment as "perfectly secure"! In a real-life scenario, every single node in a computer network is subject to a number of various security threats, and our node is no exception. Recall that to battle against different security threats we would have to employ different defence measures. As a result, a holistic approach including a collection of both reactive and proactive solutions is recommended as an effective strategy against security threats.

However, it would clearly be impracticable, if not possible, to propose a model-based solution where all types of security threats and measures were considered. Here we, therefore, concentrate on the modelling methodology rather than how inclusive these models are or should be. We follow to realise the three SSTD models for three classes of threats and their respective countermeasures, as discussed in Chapter 3. We note that any extension or shrinkage of the model with respect to the number of

[1]At this stage, we are not much concerned about where/how these packets are generated.

security submodels should be a straightforward task since the methodology remains the same.

The Keynote *For easy and meaningful comparisons, we ensure that our individual node, throughout its developing journey, owns the same amounts of resources, qualitatively and quantitatively. In particular, addition of more functionality due to deployment of any security measures in order to tackle new types of security attacks is not going to alter the computational resources of the node in any way. In other words, while the single server of the node would be exclusively dedicated to packet transmission in a perfectly secure environment, it would now have to be shared among various tasks, from its main task of transmitting packets to serving security related tasks such as anti-virus and intrusion inspection programs, encryption, etc.*

Model Realisation We utilise the SAN formalism, one of many extensions to the well-known PN formalism, to carry out the modelling activity. We make use of its features such as modularity and reusability so that security attack/defence modules can be easily added/removed as required. As a solver, we use discrete event simulation to allow for modelling and analysis of adequately detailed representation of our computer node.

To ensure modularity in modelling, we break down the node into a number of easily manageable submodels based on their functionality. To alleviate the complexity concerning the verification and validation processes [108], submodels, to a great extent, can be designed, developed, verified and validated pretty much independently prior to their integration into a single unit to represent the entire node. Recall that we propose a separate SSTD model for each class of security threats and its countermeasure. To a great extent, these are independently implemented in separate *atomic* models using the SAN formalism. We then use the composition technique to integrate and properly configure them in order to build the nodal model representing the overall node.

The node is now composed of four SAN *atomic* submodels separately designed and developed. As depicted in Figure 4.2, there exist three SAN *atomic* submodels for three SSTD models, introduced in Chapter 3, together with a SAN *atomic* submodel for the Core which mainly represents the packet encryption and transmission as well as the scheduling of the shared server as follows.

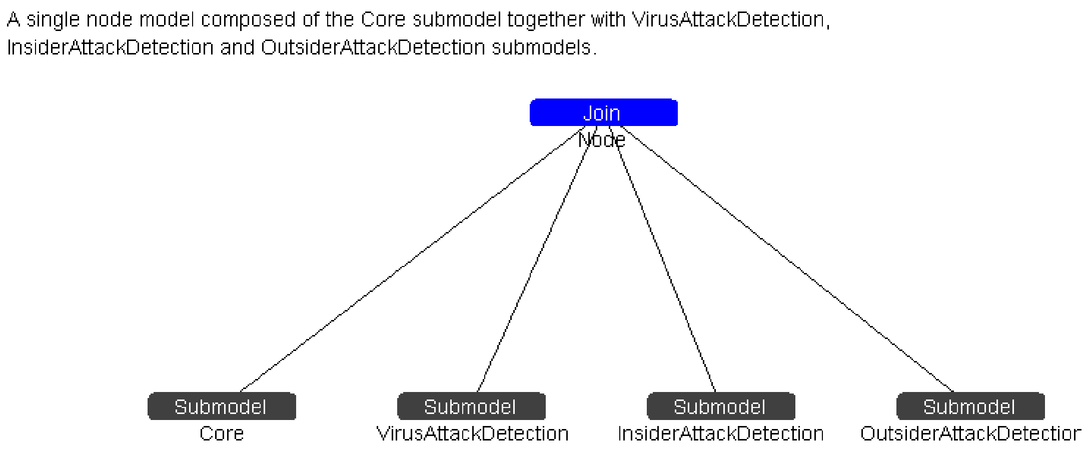
A single node model composed of the Core submodel together with VirusAttackDetection, InsiderAttackDetection and OutsiderAttackDetection submodels.
Join
Node
Submodel
Core
Submodel
VirusAttackDetection
Submodel
InsiderAttackDetection
Submodel
OutsiderAttackDetectior

Figure 4.2: The nodal model of a single server finite-capacity queueing system in a more realistic environment, represented using SAN Composed Formalism. The model combines four modularly designed SAN Atomic submodels using a Join node. The node here is assumed to be subject to three different classes of security threats: insiders, outsiders and viruses; as such, as top of packet transmission, the node employs various security measures to tackle such threats.

Here are the constituent submodels of the individual node as follows:

- *Virus Attack Detection Sub-model (VADS)*: A submodel to mimic the behaviours of virus attacks together with an anti-virus program

- *Insider Attack Detection Sub-model (IADS)*: A submodel to mimic the behaviours of insiders together with HIDS

- *Outsider Attack Detection Sub-model (OADS)*: A submodel to mimic the behaviours of outsiders together with NIDS; for clarity in design, however, the encryption mechanism, which is also part of a security measure for outsiders, is implemented in the Core submodel

- *Core*: A submodel including everything else; in particular, packet transmission, encryption process, the scheduling of the single shared server plus any *states* to be shared with the security submodels.

4.2.1 Submodels

As pointed out earlier, there are three purely security related submodels, each representing the behaviour of a certain class of security attacks together with their respective countermeasure. More specifically, the IADS captures the behaviours of insiders and the HIDS countermeasure which are commonly employed to cope with the threats originated from inside the network. In our case, insiders would include all those who have direct access to the node and intend to use it maliciously. The OADS mimics the behaviours of outsiders and that of NIDS countermeasure. The NIDSs are usually employed to deal with the attacks originated from outside the network, in this particular case, outside the node which includes the rest of the network too. The last security submodel is the VADS, capturing the behaviours of viruses and those of the anti-virus package.

Before we delve into the details of each submodel, here come some generic aspects which apply to all these security submodels in the hope that they help appreciate the modelling concepts in the following sections.

- **Nomenclature**: The three security submodels studied in this thesis have quite a lot in common. To a large extent, that is due to similarities in their definitions and functionality, which is expectedly extended to the use of nomenclature for various places (states), activities (events), input/output gates, etc. For consistency, we ensure that all similar components are given the same names as long as their scope is limited to a certain submodel. If, however, we need to share some of them in the Core submodel, we make use of an extra capital letter in front of such components to indicate the submodels they belong to. More specifically, we use **V**, **I** and **O** to refer to **V**irus, **I**nsider and **O**utsider submodels, respectively.

- **Tokens**: Each submodel utilises two tokens of which one is used to keep track of security state from perspective of that particular attack type, and the other is used to ensure requests for security inspections are made on a regular basis.

- **Miscellaneous**: The **D**etection **C**ontrol (**DC**) and the **A**ttack **C**ontrol (**AC**) places, present in each submodel, are solely for the sake of flexibility in controlling whether the submodel is configured to capture a security functionality (security attack, detection mechanism or both) or not. That is, any time we wish to switch an attack or detection mechanism on/off, we simply need to deposit/remove a token to/from the respective place, as appropriate, without having to make major changes to any other parts of the submodels. They have proved very useful, in particular during development periods. Moreover, for consistency, these also have the submodel initials in their names, e.g. VDC, VAC!

It is worth noting that the detailed quantitative modelling of security attacks and the respective countermeasures discussed below are solely based on the conceptualised and formulated SSTD models provided in Chapter 3. Any amendments in either the concepts or the models would expectedly require the respective other to be appropriately updated.

The Virus Attack Detection Submodel (VADS)

VADS mimics, in an abstract level, the behaviours of the threats introduced by viruses and the way anti-viruses react and work towards stopping them. Figure 4.3 shows the SAN *atomic* model of the VADS in full detail. More specifically, the right half (roughly) of the submodel represents the behaviour of the attacker (a virus) and the left half captures the functionality of a security measure (an anti-virus program). Below is given a brief explanation of how the model is constructed and works as follows.

- **Attack**: Recall that, by default, the system always starts from the *Secure* state. To realise this, a token is deposited to the *VSecure* place which enables the *VLearntV* activity, implying that the attacker has begun to learn more about the system and its potential vulnerabilities.

- **Detection**: The *VScanTrig* activity is ensuring that requests for security inspection (virus scan) are made on a regular basis, determined by value of the *Mean Inter-Inspection Time (MIIT)* parameter. Every time the *VScanTrig*

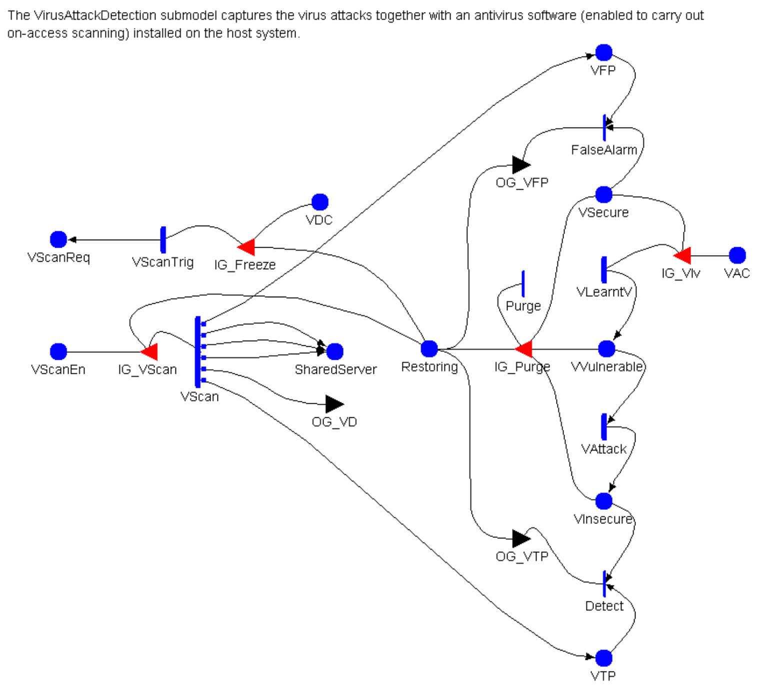
The VirusAttackDetection submodel captures the virus attacks together with an antivirus software (enabled to carry out on-access scanning) installed on the host system.

Figure 4.3: The Virus Attack Detect Submodel (VADS), represented using SAN Atomic Formalism. This mimics the behaviours of viruses (right) and those of anti-viruses (left) and their respective impcats on the node to help quantitatively examine performance implications of security, and vice versa.

activity fires, one token is deposited into the *VScanReq* place indicating that it is high time the virus inspection began although the process may have to be delayed for various reasons depending on the current state of the system.

- **Detection**: Once a request for inspection is made, subject to the (shared) server availability, the *VScanEn* is deposited with a single token, while the one in the *VScanReq* place is removed to ensure the inspection period. We note that a token in *VScanEn* enables the *VScan* activity, implying that the security mechanism (anti-virus) is now inspecting the system.

- **Detection**: If the inspection process concludes a threat despite system being in the *Secure* state, the model translates this into a FP alarm by depositing a token into the *VFP* place. This together with *VSecure* enable the instantaneous *FalseAlarm* activity which, upon firing, forces the system into unnecessary recovery!

- **Attack**: Given that a vulnerability is discovered, the *VLearntV* activity fires upon which the token is removed from the *VSecure* and deposited into *VVul-*

nerable; this announces the change of security state. As a result, the *VAttack* activity is enabled, implying that the attacker is planning to commit an active attack but has not succeeded in doing so yet.

- **Detection**: Meanwhile, if the inspection process (anti-virus) is executed, there is a possibility that the vulnerability is detected and fixed prior to any actual exploitation in which case, the system is brought back to the *Secure* state without any damage. Otherwise, the attacker is given more chance to succeed.

- **Attack**: Suppose that the virus wins this competition; that is, the *VAttack* activity fires first. That removes the token from the *VVulnerable* place and deposits it into the *VInsecure* place, implying that an active attack has taken place. We note that, while the system is compromised, the admin team is not yet aware of this! This is the most serious and potentially dangerous state. The longer the sojourn time in the *VInsecure* place, the more the likelihood of data breach! Minimising these periods should be one of the key design objectives that is only achievable with the help of quantitative models; otherwise it would be impractical, if not impossible, experimenting these with real systems.

- **Detection**: There is a high chance that occurrence of the incident is detected and appropriately responded during the next inspection process (anti-virus) run. If so, the *Detect* activity fires, removing the token from the *VInsecure* and depositing it into the *Restoring* place. Once in *Restoring*, the system undergoes a full recovery during which all its normal operations are frozen. We note that these periods contribute to unavailability of the system which is one of the key performance metrics. It is worth mentioning that there is also a possibility (FN) that the detection system identifies the attack as an acceptable activity and fails to catch that! In such cases, the system is exposed, even longer, to the attackers who have already gained unauthorised access to the resources!

The Insider Attack Detection Submodel (IADS)

The IADS represents, in a similar fashion, the behaviours of and the actions taken by the insiders and the HIDS, respectively. Recall that HIDSs are installed on the node and observe the activities taken place on it; this may include user activities, the attempts in accessing certain OS files, network ports, etc. Figure 4.4 shows the SAN *atomic* model of the IADS in full detail, which closely resembles that of VADS as discussed earlier, except that each submodel represents different attacking objects and their respective detecting mechanisms. To avoid repetition, we are not going to explain this submodel in any further details. It is well worth reminding though that there are significant differences between these two types of attacks and detection mechanisms, as discussed in Chapter 2. We also note that incorporation of both models simultaneously provides a more realistic representation of the system and, therefore, gives more accurate insights into the security related performance implications in real life systems.

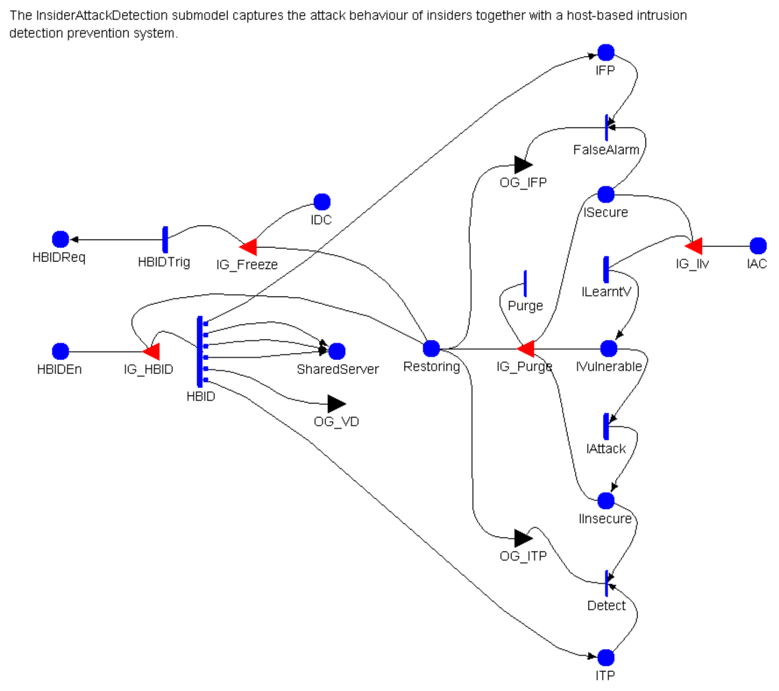

Figure 4.4: The Insider Attack Detect Submodel (IADS), represented using SAN Atomic Formalism. This mimics the behaviours of insiders (right) and those of preventive measures such as IDSs (left) and their respective impcats on the node to help quantitatively examine performance implications of security, and vice versa.

The Outsider Attack Detection Submodel (OADS)

The OADS captures the behaviours of and the actions taken by the outsiders and the NIDS, respectively. Also recall that the NIDSs are installed at some strategic points within the network in order to perform deeper packet inspections at network level. That said, the OADS should theoretically capture the collective behaviour of various security measures across the network, including firewalls, NIDS and **encryption** mechanisms, together with the behaviour of the outsider attacks, which are treated the same way.

We note, however, that different elements of this class of attack/defence mechanisms are implemented in different submodels - some in the *Core*, some in the *OADS* and some goes beyond our single node model which can be seen in the bigger picture at network level. In other words, the OADS appears slightly different from the VADS and the IADS for which all processes with regard to the security measures are executed inside the host utilising the shared server. In case of the OADS, however, only the encryption process utilises the shared server but the remaining tasks do not rely on our node resources. This is reflected by immediate activation, once a request arrives, and independent execution of the inspection mechanism in the OADS submodel, that is, without having to involve the host server. Figure 4.5 shows the SAN *atomic* model of the OADS in full detail.

Note: As we have seen earlier in Chapter 2, outsider, in general, refers to the type of attack that is originated from outside the organisations' local networks; that is, it would need to pass through various devices on the network, including firewalls, targeting a single host. Since our modelling methodology is focusing on a single node in the network, any attacks from other nodes in the network would also be considered as an outsider.

The Core Submodel

As pointed out earlier, the Core submodel plays a significant role by sticking everything together. More specifically, it captures a) the packet transmission activity,

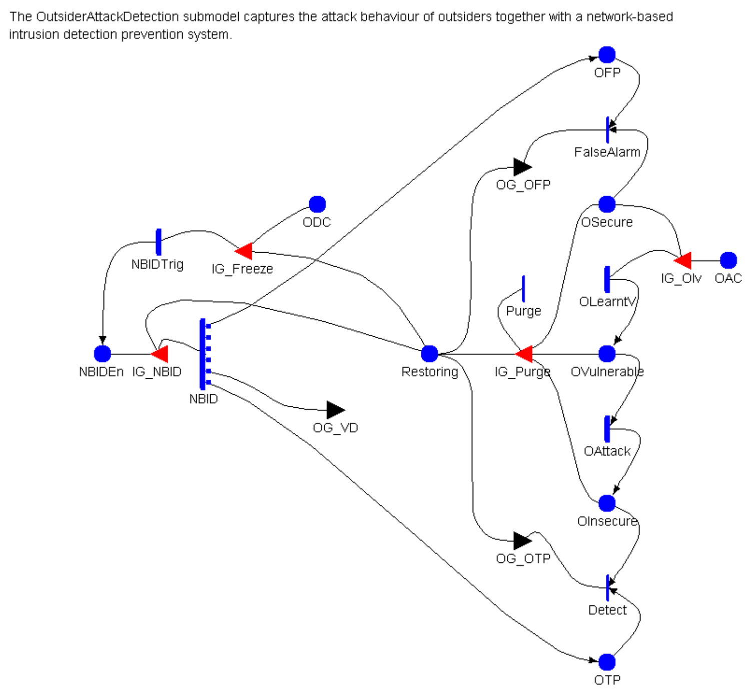
The OutsiderAttackDetection submodel captures the attack behaviour of outsiders together with a network-based intrusion detection prevention system.

Figure 4.5: The Outsider Attack Detect Submodel (OADS), represented using SAN Atomic Formalism. This mimics the behaviours of outsiders (right) and those of preventive measures such as Firewalls and IDSs (left) and their respective impcats on the node to help quantitatively examine performance implications of security, and vice versa.

which is the main functionality of the node; b) the encryption process, which is related to the OADS and acts as a preventive security measure to ensure confidentiality and integrity of the packets; c) the scheduling of the single (shared) server plus any places (*states*) which need sharing with the security submodels; and d) the arrival process. This last bit is only of interest if we wish to adopt arbitrary distribution functions for the arrival of packets.

In addition, and perhaps of paramount, the Core is accommodating for the intertwinments between performance and security, in particular, performance implications of the security related tasks. This is twofold: a) allowing the single server of the node to be shared with security measures and b) freezing all normal operation of the tasks in the event of a security failure.

Below comes in further details how the Core submodel is constructed and what the most important design factors should be considered as follows.

- **Packet Generation**: Except for recovery periods, the *PacketArr* activity is always enabled, implying that the packets are generated throughout on a

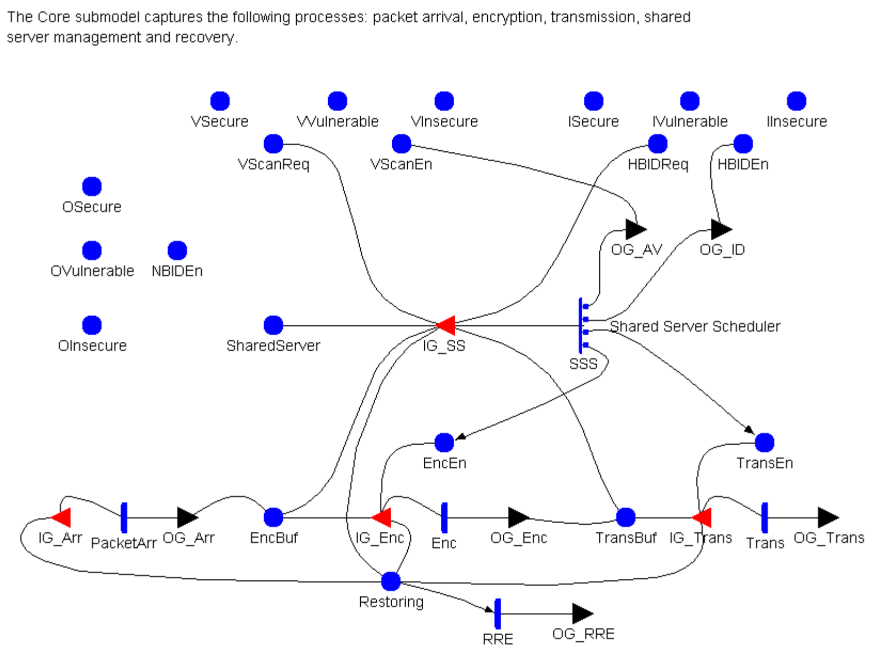
The Core submodel captures the following processes: packet arrival, encryption, transmission, shared server management and recovery.

Figure 4.6: The Core Submodel, represented using SAN Atomic Formalism. The submodel captures such features of the node as packet generation, packet transmission, packet encryption and the scheduling of the shared single server between the Core and the other security related submodels.

regular basis and according to a certain distribution function. The packets then join the *EncBuf* buffer, if possible, to be encrypted; otherwise, they are lost, contributing to the *Packet Loss Probability (PLP)* metric. It is further assumed that packet generation is not a computationally intensive process and is neglected in favour of other processes.

- **Shared Server**: The node is assumed to have a single server, as highlighted earlier, which is shared among all the main processes that are regularly running on the node. More specifically, the server provides service for virus scanning, intrusion inspection, encryption and transmission processes with their respective *activities* being *VScan*, *HIDS*, *Enc* and *Trans*. A single token in the *SharedServer* place indicates that the server is available; otherwise it is busy. We also note that the processes are non-preemptive; that is, a running process, regardless of its priority, is executed until completion. Nevertheless, the model can be flexibly adapted to accommodate for other processing disciplines if needs be.

- **Priority**: In the event of competition for the shared server, the priority is first given to the virus scanner and then to the *HIDS*; in the absence of both, an input parameter is employed to arbitrate between encryption and transmission to address any possible conflict. We use the input gate *IG_SS* together with the instantaneous activity *Shared Server Scheduler (SSS)* to manage the existing priorities and the arrangements to utilise the shared server. As expected, all these priorities can be easily rearranged to suit different application requirements.

- **Encryption**: We assume that every packet requires encryption prior to its transmission. We also follow common knowledge in assuming that longer security key lengths provide higher security levels. Recall, however, that the security key lengths and their respective encryption times are directly proportional; in other words, longer security keys require longer encryption times. To capture this notion, we virtually link the *Enc* activity in the *Core* module with the time to discover vulnerabilities in the *OADS* so that the first dynamic relation between performance and security is established.

  Although security key lengths and encryption times can be used interchangeably, in this thesis, we prefer to use encryption times due to their direct implications for performance.

- **Transmission**: Encrypted packets join the *TransBuf* buffer, subject to space availability, and wait for transmission. For the sake of simplicity, no acknowledgement or retransmission mechanisms are assumed.

- **Recovery Mode**: As discussed earlier, the overall security state is collectively monitored and updated by three security measures working collaboratively. We note that the system is considered vulnerable or insecure if either of security measures reports so; however, it is deemed secure only if it is jointly reported secure, from all security mechanisms' perspective. Once in the *Restoring* state, regardless of the cause, the *Restoration Reconfiguration Evolution (RRE)* activity is enabled, implying that the node has undergone a full recovery process. It is perfectly reasonable to assume that, during recovery, all normal functionality of the node ceases to operate; this includes the attack and detection mechanisms, packet generation, encryption and transmission. When recovery is complete, the node makes a fresh start by resetting its performance and security related states to default conditions.

4.2.2 Parametrisation

A significant number of parameters used in this research work belong to the *activities* that represent various events or actions taking place in our quantitative models. We use two types of *activities*: a) *timed activities*, which represent events that take some time to complete when enabled such as encrypting or transmitting a packet, and b) *instantaneous activities*, which represent actions that complete instantaneously (negligible compared to timed activities) when enabled such as *SSS* used to control and manage the shared server. Furthermore, every timed *activity* has a time distribution function associated with its duration. Different distributions, however, may require different sets of parameters to be fully defined. For instance, a single parameter, mean rate, is sufficient to completely characterise an exponential distribution function.

The main focus of this research thesis is on developing a methodology for the model-based quantitative analysis of performance vs security trade-offs. For the sake of simplicity, therefore, exponential distributions are chosen for all timed activities. We realise, however, that the memoryless property of an exponential distribution implies the absence of ageing and learning, which does not seem appropriate for modelling of the attackers behaviour [61]; nor does it fully capture the characteristics of a security measure. In our future work, we discuss the wide range of different distributions which are available and can be employed to capture various scenarios in a real life.

Here is a brief summary about the input parameters utilised by a number of *timed activities* as follows.

- **Mean Inter-Arrival Time (MIAT)**: This is the average inter-arrival time of packets; the inverse of this parameter would be equal to the mean rate required for the exponential distribution of the *PacketArr timed activity.*

- **Buffer Size (BS)**: There exist two finite capacity buffers in the Core sub-model. For the sake of simplicity, this single parameter determines the sizes of both buffers simultaneously. Having parametric queues allows for investigation into the impact of different buffer sizes on performance and security trade-offs.

- **Mean Transmission Time (MTT)**: This is the average time to transmit a packet; the inverse of this parameter would be the mean rate of the exponential distribution for the *Trans timed activity.*

- **Transmission Over Encryption Priority (TOEP)**: Clearly, a single shared server can only serve a single task at any given time. Therefore, a need for arbitration will arise where there are more than one task competing for the server. In the absence of security tasks, which are given higher priority, this parameter is utilised to define priority between encryption and transmission tasks. For TOEP=0.5, they are treated equally so the arbitration between them can decided with a uniform random number. If, TOEP>0.5 or TOEP<0.5, transmission or encryption would be given priority over the other, respectively.

- **Mean Inspection Time (MIT)**: This is the average time taken to execute a security inspection process; we define one parameter for both virus scanning and HIDS process times. Recall that *NIDSs* do not require the single shared server of the node to execute.

- **Mean Encryption Time (MET)**: This reflects the time required to encrypt a message, which is linked to the security key length employed by security algorithms; the longer the security keys, the longer the encryption times.

- **MIIT**: This parameter sets the time between two consecutive executions of a security measure; that is, how often we wish to inspect the node for potential intrusions.

- **FA**: This reflects the accuracy of the *IDSs* employed, namely *HIDS*, *NIDS* and anti-virus. We also assume that, for all cases, the *FP* and *FN* take equal probabilities though this might not be always desirable. Security measures may employ technologies and methods that are different from one another in which case, we may need to define two false alarm parameters, *FN* and *FP*, separately for each security measure, if required. In addition, it appears that 1% for *FN* and *FP* is deemed acceptable, reflecting the presence of a medium to high quality *IDS*.

- **Vulnerability Detection Probability (VDP)**: This indicates the probability that the system administrators will take action to patch any known vulnerabilities so that they can manage to take the system to the *Secure* state before vulnerabilities are exploited.

- **MTTOV**: This is the average time an attacker takes to discover a vulnerability in the system. It is common knowledge that the longer the security keys, the longer the time to discover vulnerabilities (the more secure the system).

- **MTIR**: This is the average time the node takes to fully recover from a security incident.

- **MTTMV**[2]: This is the average time the node takes to mitigate the known vulnerabilities and ensure the node is once again secure. Please note that this parameter works together with **VDP** to decide whether or not a known vulnerability can be mitigated on time.

- **MTTID**[2]: This is the average time the node takes to discover/detect a security incident.

- **MTBSI**[2]: This is the average time between consecutive security incidents; this parameter is not directly used in our current models.

[2]These are standardised security metrics defined by CIS, which are not directly used in the latest models introduced in this thesis. This is mainly due to the fact that the overall security is controlled by employing three different security mechanisms independently and each of which has its own SSTD with their own three states of being secure, vulnerable or insecure. In addition, some of these parameters are now implemented in the security measure model which, we believe, provides more realistic view of the real life systems. Another point to make is that some of the

4.2.3 Obtainable Measures

The Möbius tool together with the SAN modelling formalism enable construction and collection of a large set of performance values. While it is also possible to collect results for transient time analysis, the following measures are all collected in a steady-state mode. Careful observations and interpretations of these metrics can help identify and obtain invaluable insights into potential issues with respect to both performance and security. Here are the metrics we have used in this thesis as follows.

- **ssu** is a key performance metric since four crucial processes rely on the shared server availability. A server bottleneck can lead to significant degradation of performance as well as poor security in the system under consideration.

  While, throughout this research work, we mostly refer to performance implications of security related tasks, it is equally important to realise that any performance issue, e.g. fully utilised server, can lead to security implications due to the scarcity of computational resources, hence intertwinements between performance and security.

- **eu** & **tu** show what fractions of **ssu** the shared server has been busy doing encryption and transmission, respectively.

- **el** & **tl** provide estimations for the average number of jobs (packets) in the encryption buffer (*EncBuf*) and transmission buffer (*TransBuf*), respectively.

- **ear** indicates the effective (actual) packet arrival rate. There are two reasons why this might be different from the theoretic one of $1/MIAT$: a) there are times when all operations, including packet generation, are ceased; b) some packets may be lost prior to any processes (due to congestion at the encryption buffer). While this measure can provide useful information re the node itself, we use it to work out an indirect metric, *pte*, demonstrating the overall efficacy of the node.

parameters listed above are shared by three security measures. This, however, can be resolved and extended to provide more flexibility by defining distinct parameters for each measure, if needs be.

- **sth** provides the throughput, the rate at which packets are transmitted by this node. We note that any packets transmitted during the system being insecure (compromised undetected) are in fact deemed wasted, and hence excluded. It is worth mentioning that this is only checked from outsider's perspective since the encryption mechanism is considered a preventive measure only against outsiders. The **sth** metric is actually believed to be a combined metric since there is a security related condition influencing its ultimate value, as otherwise it would be different.

- (**Virus**, **Insider** & **Outsider**) Secure Probabilities indicate the fraction of time the node spends in the *Secure* state from virus', insiders' and outsiders' point of view, respectively. That is the fraction of time it can be seen in the respective places of *VSecure*, *ISecure* and *OSecure*.

- (**Virus**, **Insider** & **Outsider** Vulnerable Probabilities indicate the fraction of time the node spends in their respective *Vulnerable* state, namely in the *VVulnerable*, *IVulnerable* and *OVulnerable* places.

- (**Virus**, **Insider** & **Outsider**) Insecure Probabilities indicate the fraction of time the node spends in the respective *Insecure* state, namely in *VInsecure*, *IInsecure* and *OInsecure* places; that is, the periods when the system is compromised but undetected! These demonstrate the most serious and dangerous fractions of system life cycle which need to be minimised.

These probabilities may provide valuable insights into various aspects of the system under consideration, including behaviour of the attackers and the design characteristics and configuration settings of the security measures. They may as well make sense when considered in conjunction with other metrics. For instance, a higher sojourn time in a *Vulnerable* state may indicate: a) the system attracts attention of too many malicious hackers, b) there are too many vulnerabilities (of various types) which are probably easier to discover too, c) the system administrators are too slow in applying vulnerability patches, or d) the transition from *Vulnerable* to *Insecure* is not that straightforward, and attackers still need significant efforts to succeed despite having managed to discover some vulnerabilities. To find out the most likely reason, therefore, we may well require to consider some other factors and metrics in order to make sensible judgements.

- **rp** is the probability of the node being in the *Restoring* state; that is, the fraction of time the node is unavailable.

- System (**Secure**, System & **Insecure**) Probabilities indicate the fractions of time the node as a whole is secure, vulnerable or insecure, respectively. In order for it to be considered secure, all security measures in place must, at the same time, confirm secure states; that is, we take the joint probability (intersection) of *vsp*, *isp* and *osp*. Needless to say that the weakest link in the chain of system security would dictate the level of security in the system.

  However, being in either of insecure states regardless would suffice to announce that the system is insecure; that is the union of *vip*, *iip* and *oip*. And finally, the node will be considered vulnerable if it is not insecure but there is at least one known vulnerability. In other words, the security state of the system is in none of *vip*, *iip* or *oip*, and that it is confirmed to be at least in one of *vvp*, *ivp* or *ovp* states.

- **plp** gives the probability of packets being lost overall in the node; that is, the fraction of time that the encryption buffer, the transmission buffer, or both are full.

- **cpsm1** is the sum of *ssp*, a pure security metric, and *sth*, a performance and security metric. It combines takes contributions from both performance and security. Both contributing measures are "higher-better" [40, 112] measures and, therefore, their sum is also a "higher-better" measure.

- **cpsm2** is yet another combined metric comprised of *plp*, a pure performance metric, and *svp*, *sip* and *rp*, which are pure security metrics. The main difference is that all contributing metrics are "lower-better" [40] measures and, therefore, their sum should also be a "lower-better" measure, which we desire to minimise.

Chapter 5

Numerical Experiments and Interpretations

In this chapter, we utilise simulation modelling to demonstrate the impacts of security measures and security threats on performance of an individual computer node. We analyse the empirical data obtained from numerous simulation experiments and provide physical interpretations into optimisation of trade-offs between performance and security. We are, in particular, interested in system parameters that would optimise combined performance and security trade-offs and whether/how different input parameters could possibly influence such optimal responses.

5.1 Simulation Setup

This section describes the model and configurations under which we run the simulation experiments and collect the numerical results. In the following subsections, we briefly present the model employed and discuss main assumptions made. We also provide a procedure which summarises the methodology we follow during this simulation study.

5.1.1 The Model

A more realistic model to use would be the one we propose in Chapter 4, which captures various aspects of a holistic approach to secure an individual node by taking into consideration a number of attack scenarios together with their respective security measures. However, due mainly to severe security data scarcity, finding sensible input parameters for all these different aspects of security attacks and security measures has proved impracticable. Therefore, for the rest of this chapter, we use a simplified model representing an individual node that: a) is solely subject to attacks from outsiders; b) employs an encryption mechanism to encrypt all packets prior to their transmission; and c) is facilitated by an NIDS to identify and prevent intruders from the outside world targeting it.

The Core and OADS submodels are redrawn below in figures 5.1 and 5.2, respectively. Interested readers are referred to Chapter 4 for further details regarding these submodels.

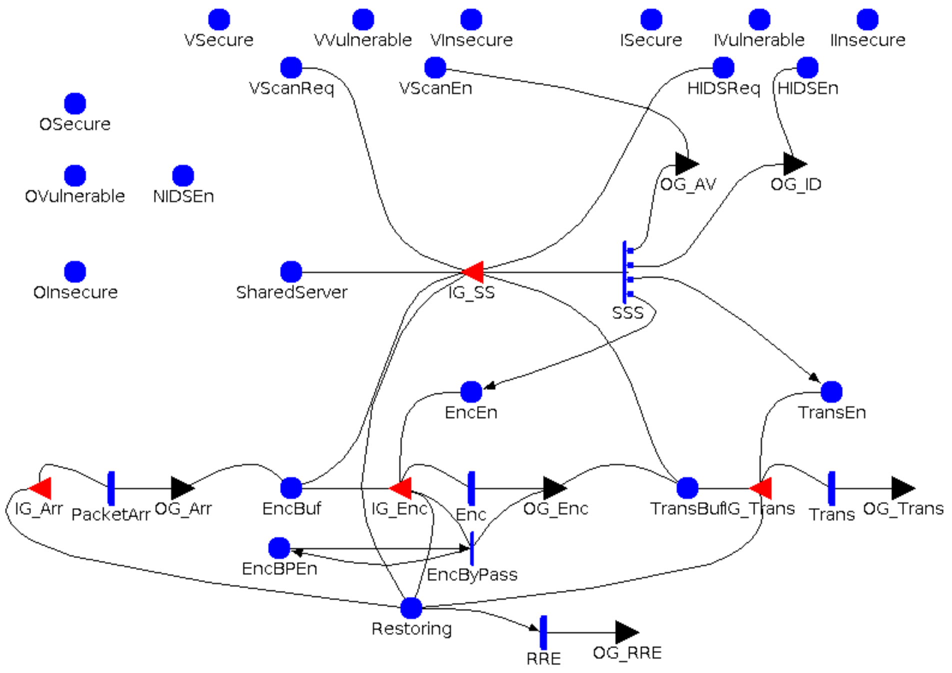

Figure 5.1: The Core Submodel

Input Parameters are some global variables that are used to parameterise the design and development of the models. In particular, they are used for characterisation of probability density functions (pdf) employed in the models and facilitate optimisation studies as and where needed.

Although other probability density functions could be easily adopted, we use Ex-

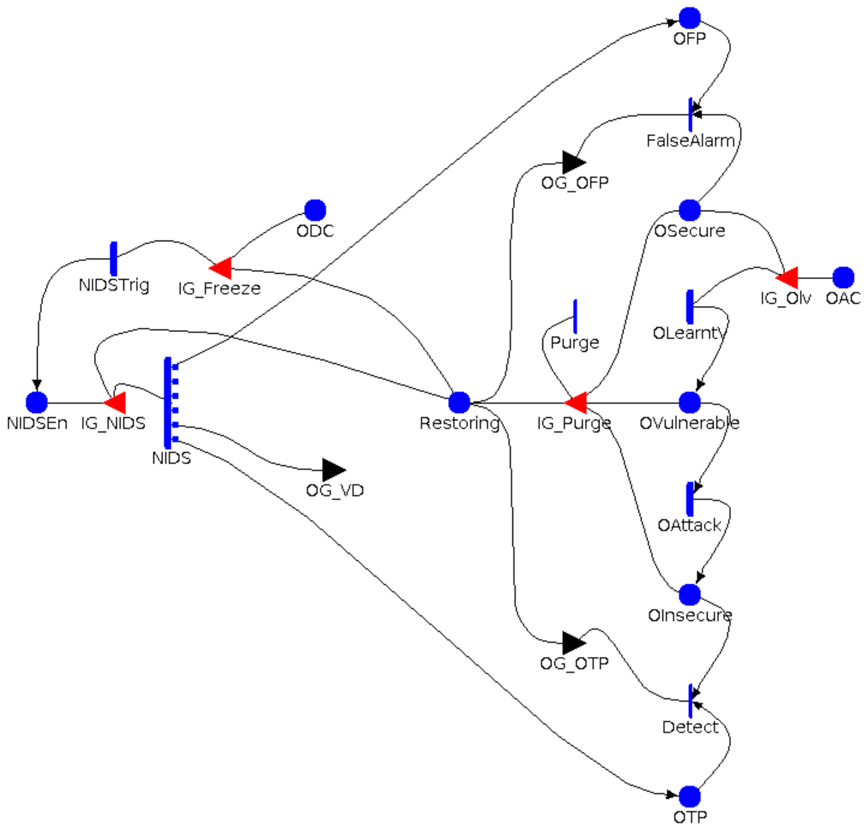

Figure 5.2: The Outsider Attack Detect Submodel (OADS)

ponentially distributed functions for all *timed activities*, where a single mean rate parameter (μ) would fully characterise each of these probability functions. Table 5.1 provides a selective list of input parameters together with some possible values attempted during various simulation runs to follow below.

Name	**Type**	**Range Type**	**Value(s)**
BS (Buffer Size)	short	Manual	[10, 30, 50]
MIAT (Mean Inter-Arrival Time)	double	Manual	[1, 1.5, 2]
MTT (Mean Transmission Time)	double	Manual	[0.1, 0.5, 1]
MET (Mean Encryption Time)	double	Incremental	[0.1, 0.2, 0.3, ..., 3.3, 3.4]
MTTOV (Mean Time To Outsider Vulnerability)	double	Manual	[12.5, 25, 50, 100, 600, 1100, 1600, 2100, ..., 14600, 15100]
FA (False Alarm)	double	Manual	[0, 0.005, 0.01, 0.02]
MIIT (Mean Inter-Inspection Time)	double	Manual	[10, 30, 50, 100, 200, 400]
MIT (Mean Inspection Time)	double	Manual	[0.1, 0.5, 1]
MTTR (Mean Time To Recovery)	double	Fixed	[360]
TOEP (Transmission Over Encryption Priority)	double	Fixed	[0.5]
VDP (Vulnerability Detection Probability)	double	Fixed	[0.8]

Table 5.1: Input parameters defined and used in the simulation models

Performance Variables are reward variables recorded during simulation experiments. They are defined as either Rate Rewards or Impulse Rewards; the former being based on the time spent in each state, the latter defining functions that are evaluated when activities in the child models are fired. The Impulse Rewards are mostly used as counters to work out the average number of times the respective activities fire. Moreover, all these reward variables are defined as Steady State response types. That is, the reward functions are evaluated after the system being modelled reaches steady state.

While there exist other types of estimates, we collect estimates for mean of these reward variables. And finally, the confidence level across all simulation runs and for all reward variables is set to 95%. Despite this setting, however, it is likely that some performance variables in certain experiments are unable to satisfy the confidence level requirement, e.g. due to limited number of batches used in the experiments. If so, an asterisk (*) is added next to such variables to indicate the occurrence of dissatisfaction. Table 5.2 provides a selective list of performance variables of interest in this numerical study.

Name	**Type**
ssu (Share Server Utilisation)	Rate Reward
eu (Encryption Utilisation)	Rate Reward
tu (Transmission Utilisation)	Rate Reward
el (Encryption Queue Length)	Rate Reward
tl (Transmission Queue Length)	Rate Reward
ear (Effective Arrival Rate	Impulse Reward
sth (Secure Throughput)	Impulse Reward
rp (Prob. of System Found in Recovery State)	Rate Reward
ssp (Prob. of System Found in Secure State)	Rate Reward
svp (Prob. of System Found in Vulnerable State)	Rate Reward
sip (Prob. of System Found in Insecure State)	Rate Reward
plp (Packet Loss Prob.)	Rate Reward

Table 5.2: Performance variables defined and used in the simulation models

5.1.2 Combined Performance and Security Metrics

We refer to the above response variables as individual reward functions; that is, they are either performance or security related variables. The combined metrics, on the other hand, are defined such that aspects of both performance and security can at the same time contribute to. More specifically, we define two combined metrics as follows:

- **cpsm1** $=$ **ssp** $+$ **sth**: While **ssp** is a pure security metric that gives the probability of the node being secure (in *Secure* state), **sth** is a performance metric influenced by security state of the node. As mentioned earlier, **sth** gives secure throughput, the average packet transmission rate subject to system being secure. In other words, those packets transmitted during insecure periods (**cu**) are excluded. Both **ssp** and **sth** are "higher-better" metrics and therefore **cpsm1** also is a "higher-better" metric.

- **cpsm2** $=$ **svp** $+$ **sip** $+$ **rp** $+$ **plp**: The first two components, **svp** and **sip**, may be considered as pure security metrics, whereas **plp** is a mixed performance and security metric. It may also be argued that **rp** can be seen as a performance measure, in the form of unavailability; we, however, realise that seeking to make computer resources unavailable is one major objective of security attacks[1]. All components of **cpsm2** are 'lower-better' metrics and, therefore, **cpsm2** should be a 'lower-better' metric.

5.1.3 Assumptions

First and foremost, we ensure that the computational capacity of the individual node remains the same throughout all simulation runs and experiments (as a control variable). That is, addition of more functionality, due mainly to the employment of security measures, is not going to alter this capacity in any way. In other words, while the single server of a perfectly secure node would exclusively be serving packet transmission, it would, in a more hostile environment, have to serve security related tasks as well. This allows for more sensible comparisons to be made and meaningful conclusions drawn later on.

Furthermore, all experiments to follow are based on the fact that security of an individual node varies with respect to the length of security keys employed. As commonly agreed, longer security keys require, on average, relatively more time to

[1]Recall that the ultimate goal of security is to protect confidentiality, integrity and availability

break, hence improved security. However, longer keys are translated into longer encryption times, implying that they are likely to lead to significant performance implications too. As detailed in Table 5.1, simulation runs are executed for *34* different security key lengths where each key is associated with a different encryption time. Moreover, depending on the length of the security key employed, the time to vulnerability for outsiders (level of security) varies such that longer keys lead to longer times before a vulnerability can be discovered.

5.1.4 Design of Experiments

There exist relatively a large number of input parameters that can possibly influence the outcomes of simulation experiments. The total combinations of various parameters can easily explode far beyond anything practical. For instance, there would be over 10^9 experiments (cartisian product) to run if each input parameter was only to take on 4 different values, still not representative of reasonably high dynamic operating conditions. Therefore, careful design of simulation experiments is crucial and can save a lot of time and effort by providing efficient ways to estimate the effects of changes in the model's inputs on its outputs. Kelton [46] raises a number of questions that need to be answered before just trying different things to see what happens. They are as follows:

- What model configurations should we run?
- How long should the runs be?
- How many runs should we make?
- How should we interpret and analyse the outputs?
- What's the most efficient way to make the runs?

Some of these questions are dealt with when we design and develop simulation models in Chapter 4, where we decide what performance variables to include, how to define them, what configurations of input parameters should be considered and why, etc. Others are addressed in this chapter by considering, for instance, a) our particular interest in steady-state rather than transient simulation types; b) a suitable confidence level (CL) to terminate a given experiment, say 95%; c) some variance reduction techniques to possibly sharpen the precision of output estimators without having to do more simulations [46]; and d) the impact of certain input parameters on outputs (sensitivity analysis).

We adopt an incremental methodology to conduct this numerical study. That is, we begin with a minimal model as a reference and then, to increase model complexity, add a single new functionality to it each time. Here is the procedure in further detail as follows:

- We design a simplest possible model configuration that serves as the basis of our comparisons throughout. This reference model represents a single server node that would be operating in a perfectly secure environment. The main advantage of this model is that there exist exact solutions [4] that help validate our numerical results in the very first stage.

- Despite the node still operating in a fully secure environment, we go ahead with our incremental development and employ an encryption mechanism so that all packets would be encrypted prior to their transmission. This added functionality should impose extra load onto the single server node. We plan to investigate the performance implications of this first security mechanism before it becomes more complicated.

- Next comes a further developed model to represent the node in a more realistic operating condition. That is, the node is now subjected to security attacks from outsiders and, to cope with potential intrusions, it needs to employ further security measures. As expected, these new features should impose undue computational and network overhead; nevertheless, we assume the node retains its original computational capacity so as to allow for more sensible comparisons.

- We then fix all input parameters, except for MET and MTTOV, at some reasonable values. The values of MET and MTTOV are associated with the security keys employed and as we vary the key length from experiment to experiment, the MET and MTTOV are also accordingly changed. For a set of fixed input factors, we sweep through a range of different key lengths, run the experiments and collect various performance variables of interest. We can then plot the responses with respect to MET[2] at the end of each run to analyse the behaviour of each response.

- We realise that not all input factors are equally important in terms of having a major impact on the outputs. There are several factor screening designs in literature [10, 38, 39, 48, 53] that are extremely helpful in transforming a rather hopelessly large number of runs into something that is eminently manageable. Here, however, we are very selective about our input parameters as we solely aim to illustrate how different configurations could significantly change the optimal trade-offs (responses) between performance and security. Therefore, we simply choose a set of input factors that we believe are important; these include BS, FA, MIAT and MIIT.

- Last but not least, we ensure that, at any given experiment, only one of these input parameters (along with the key length) is varying so that we can draw more accurate and reasonable conclusions as to whether the impact of that input factor on various responses is significant.

5.2 Simulation Runs

Following the methodology discussed above, we design and run a number of simulation scenarios. Generally, we aim at investigating the implications of security measures and security attacks for performance of a single computer node. In particular, we wish to:

- Identify the security keys for which optimal trade-offs between performance and security can be achieved. That is, any security key that can maximise/minimise

[2]The actual independent variable here is the length of security key and the MET is assumed to be a monotonically increasing function of security key length. Nevertheless, it is easier and more sensible to work with the latter than the former

our combined metrics of cpsm1/cpsm2, respectively.

- Examine the impact of various input parameters on these optimal trade-offs. Of particular interest, are BS, MIAT, MIIT and FA parameters.

5.2.1 Perfectly Secure Scenario

This scenario can in fact represent an extreme case where an embedded system is connected to a secure, reliable and fully isolated local network [112]. There would be little, if any, security concerns for such a system and therefore the server would be expected to work towards and achieve its near nominal performance. Although making such presumptions about security of a computer node may sound unrealistic, this scenario is aimed at providing a basis for more sensible comparisons.

To ensure a smooth transition, here we consider two separate cases, **with** and **without** encryption mechanisms, as follows.

Without Encryption Mechanism

We begin with an individual node where security measures are **all** made redundant. Such a node would solely be responsible for transmitting packets without possibly being disrupted with any security tasks or issues. The model configuration for this case can be effectively reduced to an idealised SAN atomic model of an M/M/1/K queueing system, a single server with finite capacity[3]. (Figure 5.3)

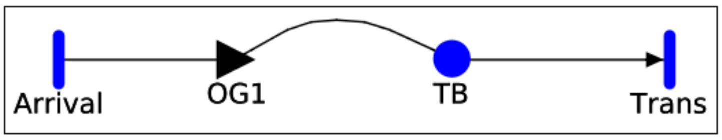

Figure 5.3: The SAN Atomic model of an M/M/1/K queueing system

Simulation runs are executed aiming to provide insights into various reward functions defined in this simplified model. Table 5.3 below shows the input parameters

[3]In QN, an M/M/1/K notation implies a FCFS queueing discipline in place; this, however, cannot be guaranteed in SAN formalism since tokens are not distinguishable.

used during these experiments.

Name	**Value(s)**
BS	[10, 30, 50]
MIAT	[1, 1.5, 2]
MTT	[0.1]

Table 5.3: Input parameters in a perfectly secure environment where there is no encryption process nor is the node subject to any potential security threats.

Simulation results are briefly presented in three different groups of graphs: a) individual performance metrics; b) individual security metrics; and c) combined performance and security metrics. Figure 5.4 shows the results for different buffer sizes (BS), where the packet arrival rate remains unchanged.

The M/M/1/K queueing system is a well-known model with exact solutions, and thus the interpretation of all results are pretty straightforward. Here is the summary of the outcomes highlighted as follows:

- The **ear** is exactly the same as the arrival rate, implying that all arriving packets make their way through transmission. In other words, the **plp** is almost always zero and the chance of losing packets due to lack of buffer is negligible, despite finite capacity.
- Throughput, as confirmed, is therefore equal to arrival rate.
- There is no encryption mechanism in place, which is confirmed by zero utilisation of **eu** reward function.
- The shared server is only responsible for transmitting the packets, which is again confirmed by **ssu** being always equal to **tu**.
- The node has no vulnerability, implying that it is always secure. As such, there is no chance of finding the node in the **Recovery** state.
- There is no mystery about **cpsm1** and **cpsm2**; as defined earlier, they are just calculated using some individual metrics. It is therefore clear from the curves that none of these combined metrics shows any optimal trade-offs for performance and security.

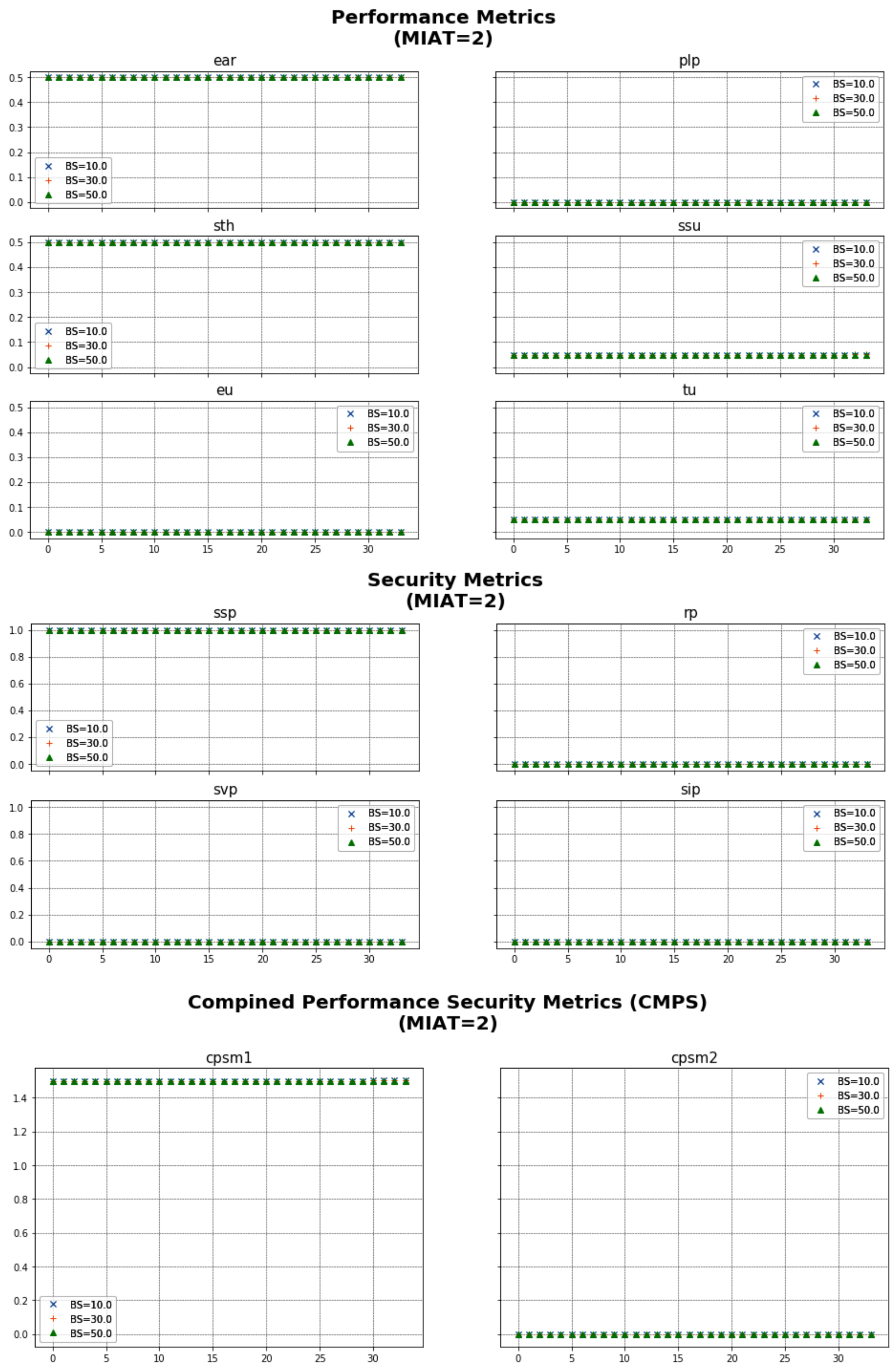

Figure 5.4: Given the node is operating in a perfectly secure environment and MIAT=2, the graphs show simulation results with respect to three different values of the BS parameter. The x-axes, shared among all graphs, simply indicate an index number with no dimension (the very same results are achieved 34 times to allow for easy comparison later on). For the top and middle groups of graphs, the y-axes indicate either probability or normalised values varying in the range of [0-1]. For the bottom graphs, however, they inidcate plain numbers (no dimension); the values here are achieved by adding corresponding values of some individual metrics making each combined metric. This is only useful for comparison as well as identification of optimum points, if any.

Figure 5.5 shows the results for a similar set of experiments except that here packet arrival rate is varying and buffer size remains unchanged throughout the runs. In brief,

- All security metrics remain the same as before since nothing has really changed from security point of view.
- Despite increased arrival rates, it is confirmed that the buffer size (BS=30) together with the server rate are sufficiently high to guarantee zero packet loss. (**plp**=0)
- The changes in the arrival rate are correctly and expectedly reflected in both **ear** and **sth** graphs. The same is for utilisation of the server since higher arrival rates would require more server times, which can be observed from **ssu** and **tu** plots.
- Again, neither **cpsm1** nor **cpsm2** exhibit any optimal trade-offs for performance and security.

With Encryption Mechanism

In this scenario, the node employs an encryption mechanism while still operating in a fully trusted environment. That is, all security related tasks except for encryption continue to remain disabled. Our single server node is now responsible for both encryption and transmission of the packets. As such, any significant changes in simulation outcomes should presumably be considered as performance implications of the encryption mechanism.

The model configuration for this scenario can be seen as though the service time of the M/M/1/K queueing system is increased such that it now takes longer to serve each packet than before. As an immediate effect, this should therefore increase utilisation of the server, which may in turn have some impacts on other metrics such as **plp** and **sth**.

It is worth noting though that in addition to controlled changes in BS and MIAT pa-

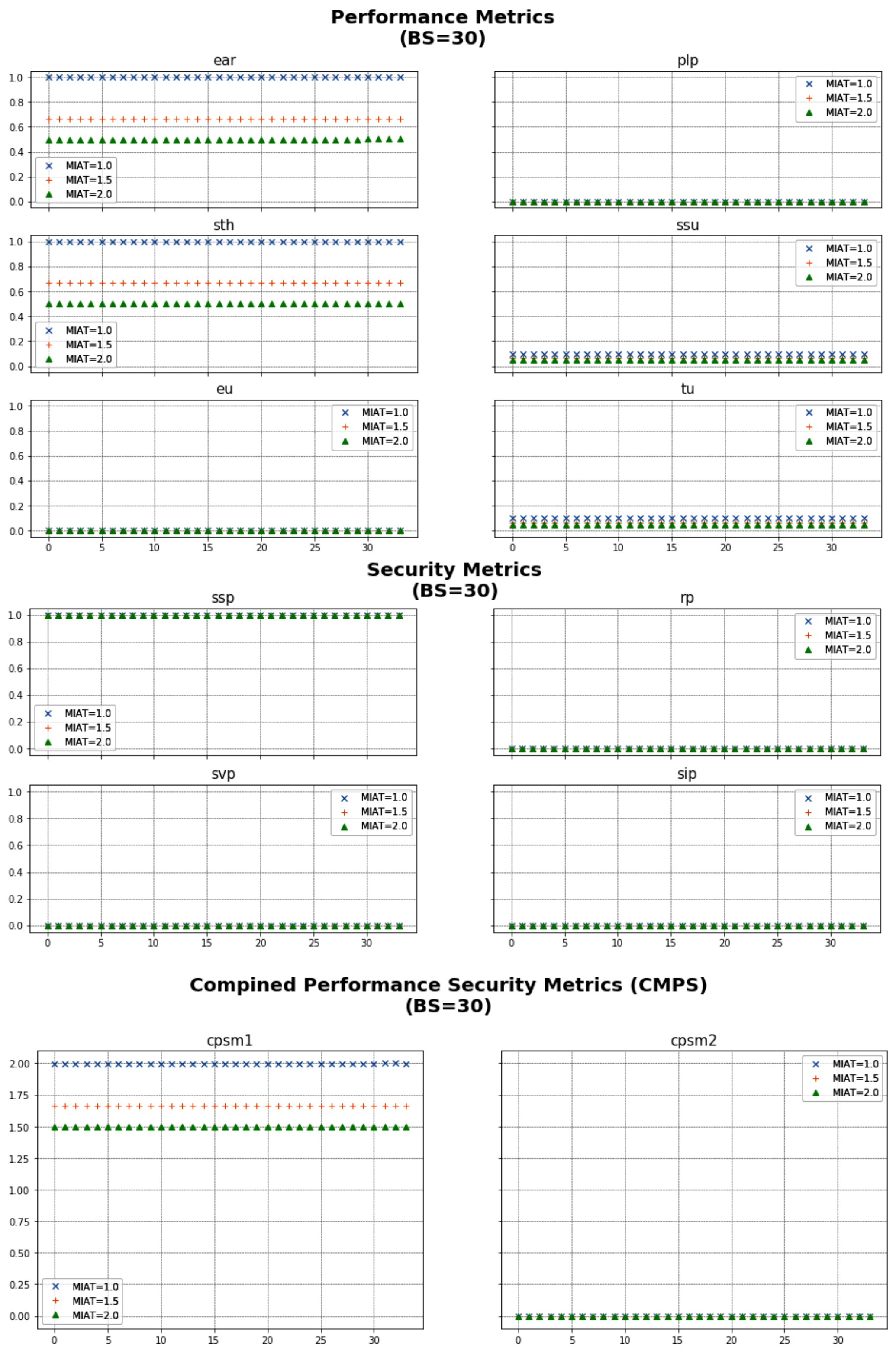

Figure 5.5: Given the node is operating in a perfectly secure environment and BS=30, the graphs show simulation results with respect to three different values of the MIAT parameter. The x-axes, shared among all graphs, simply indicate an index number with no dimension (the very same results are achieved 34 times to allow for easy comparison later on). For the top and middle groups of graphs, the y-axes indicate either probability or normalised values varying in the range of [0-1]. For the bottom graphs, however, they indicate plain numbers (no dimension); the values here are achieved by adding corresponding values of some individual metrics making each combined metric. This is only useful for comparison as well as identification of optimum points, if any.

rameters, now the security key length is also varying across the experiments, which are respectively reflected in the encryption times of the server as discussed previously.

Table 5.4 and Table 5.5 below provide the input parameters used during these experiments as follows.

Name	Value(s)
BS	[10, 30, 50]
MIAT	[2]
MTT	[0.1]
MIIT	[30]
MIT	[0.1]
FA	[0]
TOEP	[0.5]
MET	[0.1, 0.2, 0.3, ..., 3.3, 3.4]
MTTOV	[12.5, 25, 50, 100, 600, 1100, 1600, 2100, ..., 14600, 15100]

Table 5.4: Input parameters (BS varying) used in a perfectly secure environment with an encryption process in place

Name	Value(s)
BS	[30]
MIAT	[1, 1.5, 2]
MTT	[0.1]
MIIT	[30]
MIT	[0.1]
FA	[0]
TOEP	[0.5]
MET	[0.1, 0.2, 0.3, ..., 3.3, 3.4]
MTTOV	[12.5, 25, 50, 100, 600, 1100, 1600, 2100, ..., 14600, 15100]

Table 5.5: Input parameters (MIAT varying) used in a perfectly secure environment with an encryption process in place

Simulation results for different values of BS and MIAT are shown in Figure 5.6 and Figure 5.7, respectively. We realise, from Figure 5.6, that:

- The buffer size, BS, has very little influence on the responses.
- The **ear** remains unaltered regardless. This is because the link controlling the arrival process is not activated yet, and therefore the packets are generated without any interruption.
- The longer the security key length, the longer the encryption time and that. As the encryption time is increasing, the server utilisation is increasing too.
- A fully utilised server ends up losing arrival packets. This is confirmed by the **plp** graph. It further shows that the probability of loss for shorter buffers is higher, which is well understood.
- Any increase in **plp** should also be reflected in the throughput of the system.

This can be seen by the **sth** graph which begins to decrease by any further increase in encryption time.

- The severe implications of encryption for performance is clearly observable when the utilisation of encryption is compared with that of transmission. While the latter is almost negligible, it further decreases as the system loses more packets in case of longer keys.

- The security metrics are identical to those of previous scenario. This is expected since the node is not subject to any security threats yet, and as before, it is always secure.

- Although the **cpsm1**/**cpsm2** metrics begin to decrease/increase, respectively, as the key length increases, they do not show any optimal values. Clearly, the lower the security key, the less performance implications of security.

For different arrival rates, the trend continue to remain the same except that the abrupt changes in the responses are shifted to the left/right for higher/lower arrival rates. Figure 5.7 shows the simulation results, where all experiments are run for a fixed buffer size of BS=30. In brief, we realise that

- All security metrics remain the same as before since nothing has really changed from security point of view.

- Despite some higher arrival rates, the **plp** still confirms that the buffer size (BS=30) together with the server rate are sufficiently high to guarantee zero packet loss.

- The changes in the arrival rate are correctly and expectedly reflected in both **ear** and **sth** graphs. The same is for utilisation of the server since higher arrival rates would require more server times, which is observable from **ssu** and **tu** plots.

- AS before, there are no optimal points for neither **cpsm1** nor **cpsm2**.

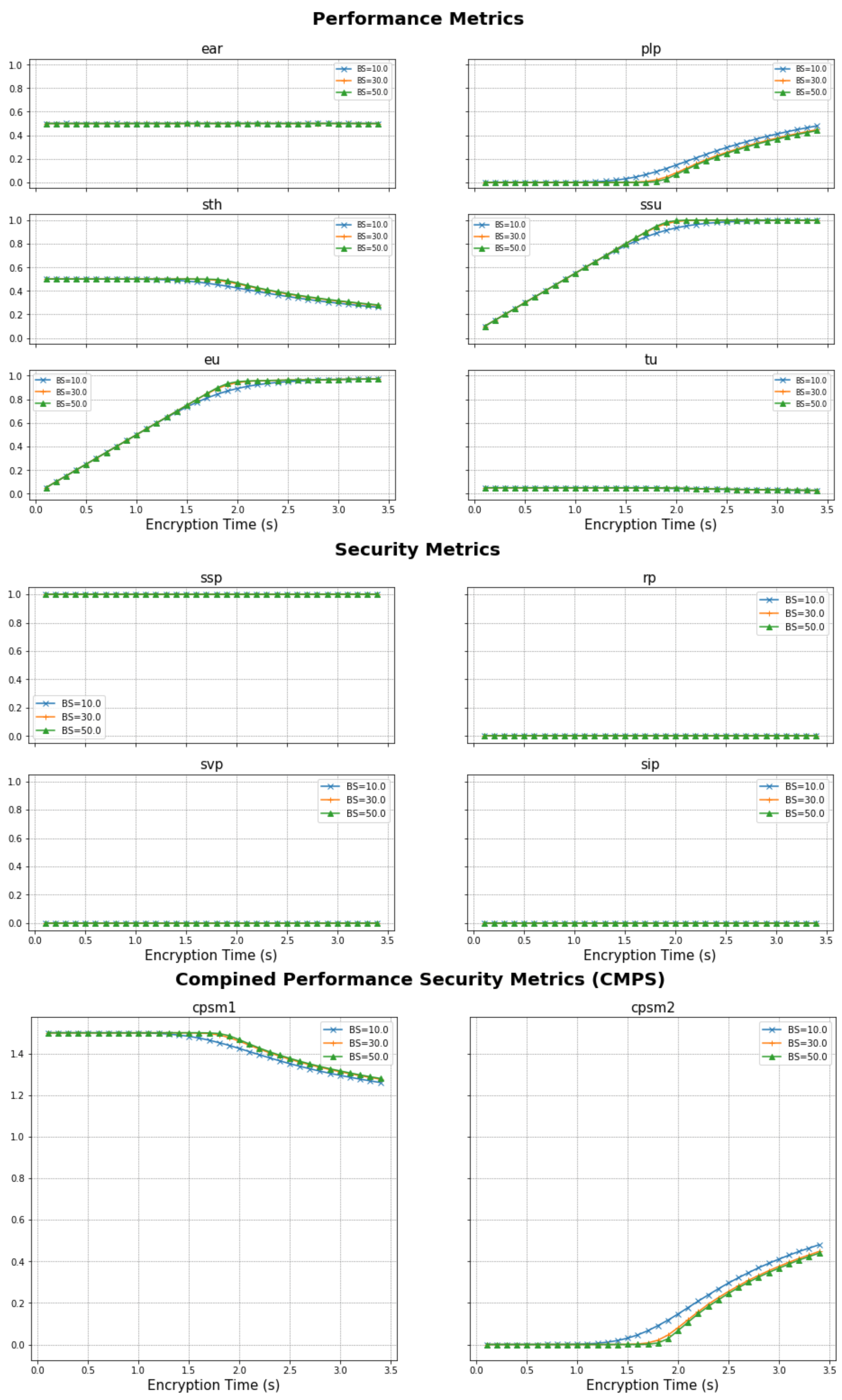

Figure 5.6: Despite a perfectly secure environment, an encryption mechanism is employed to highlight the overhead. Given MIAT=2, the graphs show simulation results for three different values of the BS parameter. The x-axes, shared among all graphs, indicate the encryption times corresponding various security key lengths (34 different keys). For both performance and security individual metrics, the y-axes indicate either probability or normalised values varying in the range of [0-1], whereas for Combined metrics they only indicate plain numbers, achieved by adding corresponding values of some individual metrics making each Combined metric.

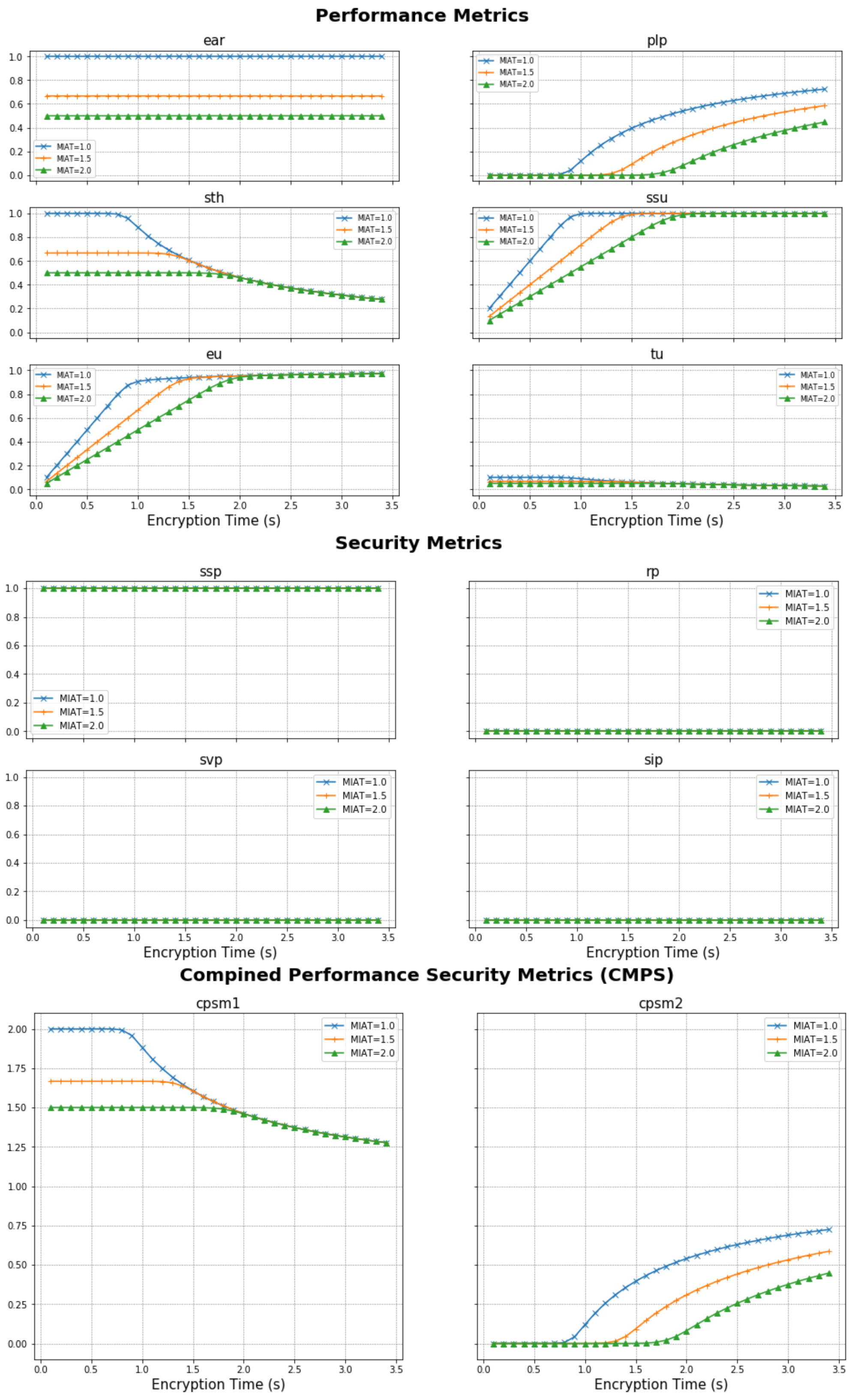

Figure 5.7: Despite a perfectly secure environment, an encryption mechanism is employed to highlight the overhead. Given BS=30, the graphs show simulation results for three different values of the MIAT parameter. The x-axes, shared among all graphs, indicate the encryption times corresponding various security key lengths (34 different keys). For both performance and security individual metrics, the y-axes indicate either probability or normalised values varying in the range of [0-1], whereas for Combined metrics they only indicate plain numbers, achieved by adding corresponding values of some individual metrics making each Combined metric.

5.2.2 Optimal Key Length Scenario

It is time to move on to a more realistic model where the node is subject to security threats from outsiders and that further security measures are employed to prevent intrusion as well as unauthorised access to the data. The model configuration for this scenario is the same as those drawn in figures 5.1 and 5.2 above, where all security related tasks are once again active and operational.

Following a similar procedure as above, simulation experiments are executed for different buffer sizes and arrival rates as detailed in Table 5.6 and Table 5.7.

Name	Value(s)
BS	[10, 30, 50]
MIAT	[2]
MTT	[0.1]
MIT	[0.1]
FA	[0]
MIIT	[30]
MET	[0.1, 0.2, 0.3, ..., 3.3, 3.4]
MTTOV	[12.5, 25, 50, 100, 600, 1100, 1600, 2100, ..., 14600, 15100]
MTTR	[360]
TOEP	[0.5]
VDP	[0.8]

Table 5.6: Input parameters (BS varying) used to identify optimal keys where the node is subject to the outsiders

Name	Value(s)
BS	[30]
MIAT	[1, 1.5, 2]
MTT	[0.1]
MIT	[0.1]
FA	[0]
MIIT	[30]
MET	[0.1, 0.2, 0.3, ..., 3.3, 3.4]
MTTOV	[12.5, 25, 50, 100, 600, 1100, 1600, 2100, ..., 14600, 15100]
MTTR	[360]
TOEP	[0.5]
VDP	[0.8]

Table 5.7: Input parameters (MIAT varying) used to identify optimal keys where the node is subject to the outsiders

Simulation results are illustrated in figures 5.8 and 5.9, revealing very interesting insights into the node model. The key finding is that the node now becomes a bottleneck not only for longer but also for shorter security keys. While encryption mechanism is the major cause of bottleneck for longer keys, we learn that increased recovery time, due to frequent security attacks, severely impacts the performance of the system for shorter keys. Below comes more detailed interpretation of what is happening in these graphs as follows.

- Recall that all normal operations in the node are frozen during a recovery period, packet generation is no exception. Employment of shorter security keys leads to a less secure node; such a susceptible node would end up in **Recovery** state more often, hence longer frozen times overall.

- The **sth** graph now significantly drops for shorter keys. This is associated with the frequent denial of arrivals, which is also in line with the **ear** graph.

- Unlike previous scenarios, the node is no longer always secure; this is well understood. The security metrics are now indicating the probabilities of the node being secure, vulnerable, insecure or recovery.

- While there is no optimal (maximum) in neither **sth** nor **ssp** with respect to any security key length, **cpsm1** exhibits a clear optimal for a certain key.

- There is no optimal (minimum) for any constituents of **cpsm2** either; nevertheless, for the very same key length, **cpsm2** exhibits a clear optimal (minimum).

- We also learn that these optimal keys may vary with respect to the BS parameter; the smaller the buffer size, the shorter the optimal key.

- Although the graphs may look different, the arrival rate has similar influence as the buffer size. The higher the arrival rate, the shorter the key length.

- The BS parameter has a very similar influence on the response, though in a smaller scale. In other words, shorter buffer sizes tend to shift the optimal trade-offs towards left.

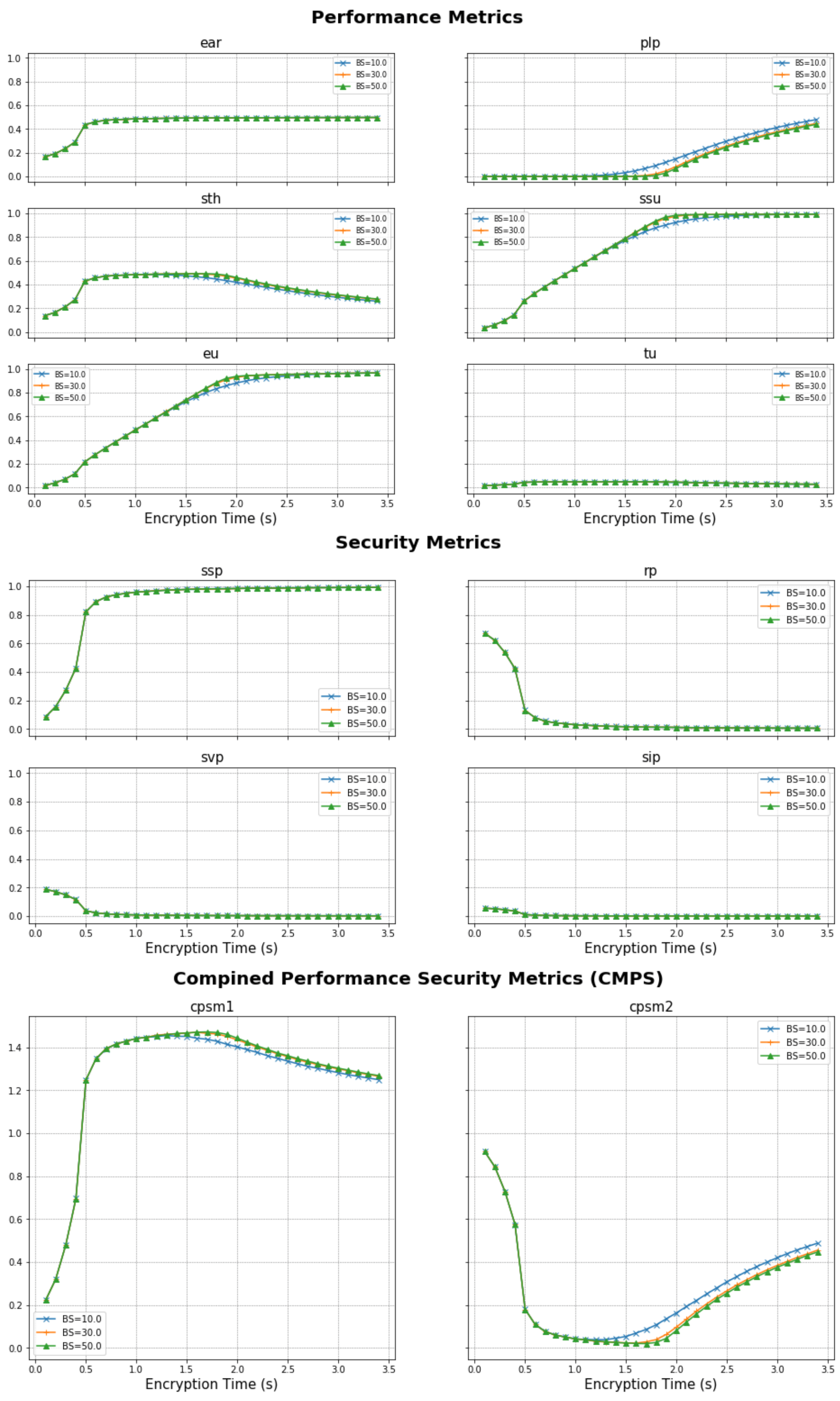

Figure 5.8: The graphs show the impact of outsider threats and the overhead resulted from an encryption mechanism employed to tackle with such threats. Given MIAT=2, simulation results are achieved for three different values of the BS parameter. As before, the x-axes are shared among all graphs, indicating the encryption times corresponding various security key lengths (34 different keys). For both performance and security individual metrics, the y-axes indicate either probability or normalised values varying in the range of [0-1], whereas for Combined metrics they only indicate plain numbers, achieved by adding corresponding values of some individual metrics making each Combined metric.

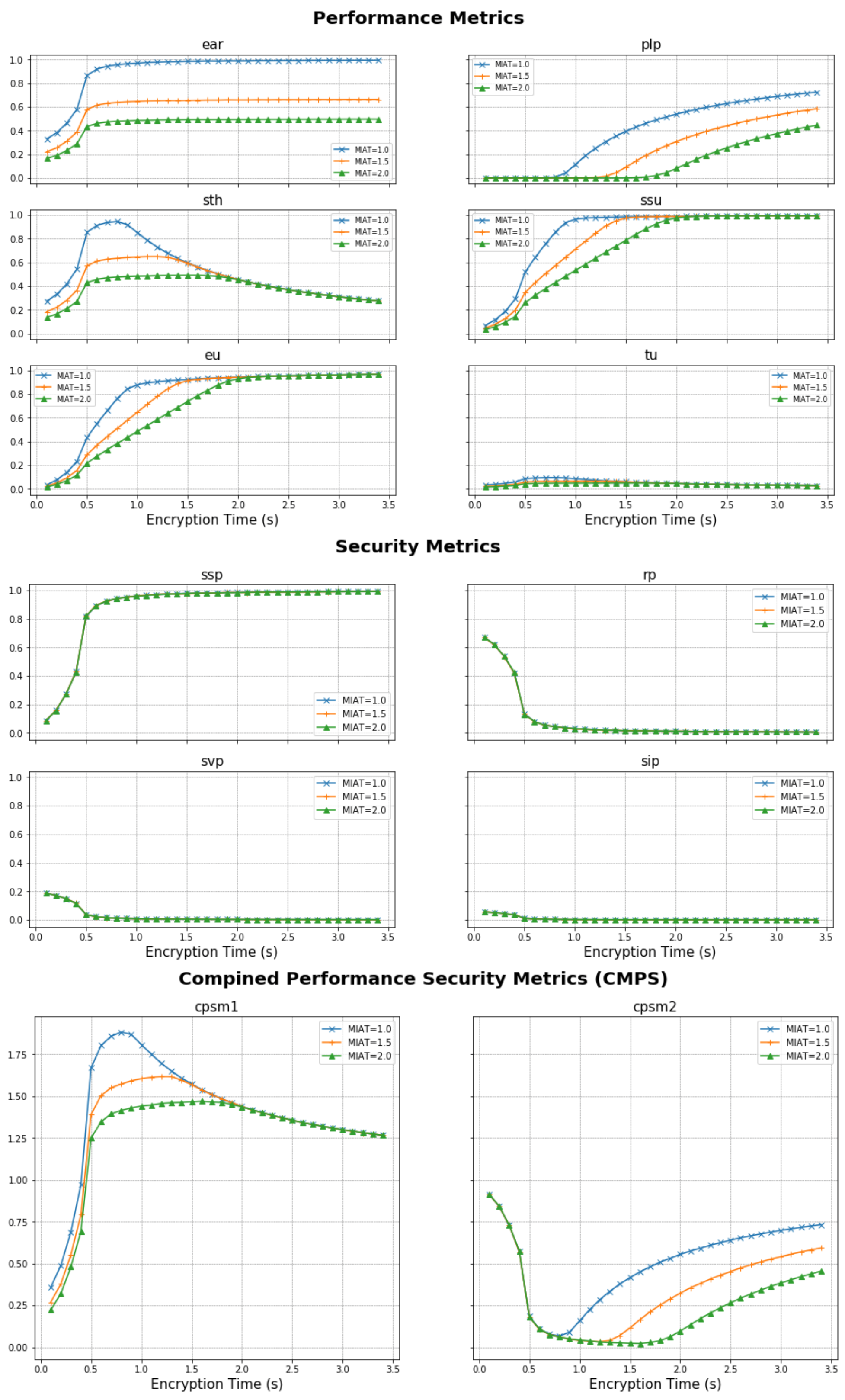

Figure 5.9: The graphs show the impact of outsider threats and the overhead resulted from an encryption mechanism employed to tackle with such threats. Given BS=30, simulation results are achieved for three different values of the MIAT parameter. As before, the x-axes are shared among all graphs, indicating the encryption times corresponding various security key lengths (34 different keys). For both performance and security individual metrics, the y-axes indicate either probability or normalised values varying in the range of [0-1], whereas for Combined metrics they only indicate plain numbers, achieved by adding corresponding values of some individual metrics making each Combined metric.

5.2.3 False Alarm Scenario

Recall that FP and FN are inherent characteristics of every security measure and putting FA=0 would be somewhat unrealistic. In this section, therefore we investigate the impacts of different FA rates on optimal keys identified in previous sections. Table 5.8 provides the list and settings of input parameters used in these models.

Name	**Value(s)**
BS	[30]
MIAT	[2]
FA	[0, 0.005, 0.01, 0.02]
MTT	[0.1]
MIT	[0.1]
MIIT	[30]
MET	[0.1, 0.2, 0.3, ..., 3.3, 3.4]
MTTOV	[12.5, 25, 50, 100, 600, 1100, 1600, 2100, ..., 14600, 15100]
MTTR	[360]
TOEP	[0.5]
VDP	[0.8]

Table 5.8: Input parameters used to identify optimal keys (impact of FA parameter)

The simulation outcomes, shown in Figure 5.10, exhibit an explicit trend with respect to FA. We observe that increasing FA rates leads to significant drops in both **sth** and **ssp**. Here is why as follows.

- Let us begin with the FA rates equal to 0, which implies the security measures perform perfectly accurately; that is, they are capable of identifying the security state of the node correctly at all times, whether it be secure or insecure state. As such, the false alarm activities (FP & FN) in security submodels never fire, delivering the most optimistic results that are solely useful for comparison purposes.

- Real life systems, however, usually come with non-zero FA rates. These may pose significant performance penalties and security concerns to computer networks, which are well worth careful investigations. Figure 5.10 shows the simulation results under various settings for FA. Here, a FA rate equal to 0.02, for instance, would imply that the security measure, in 2 out of 100 inspections on average, would either falsely signal a security attack (FP: fires the

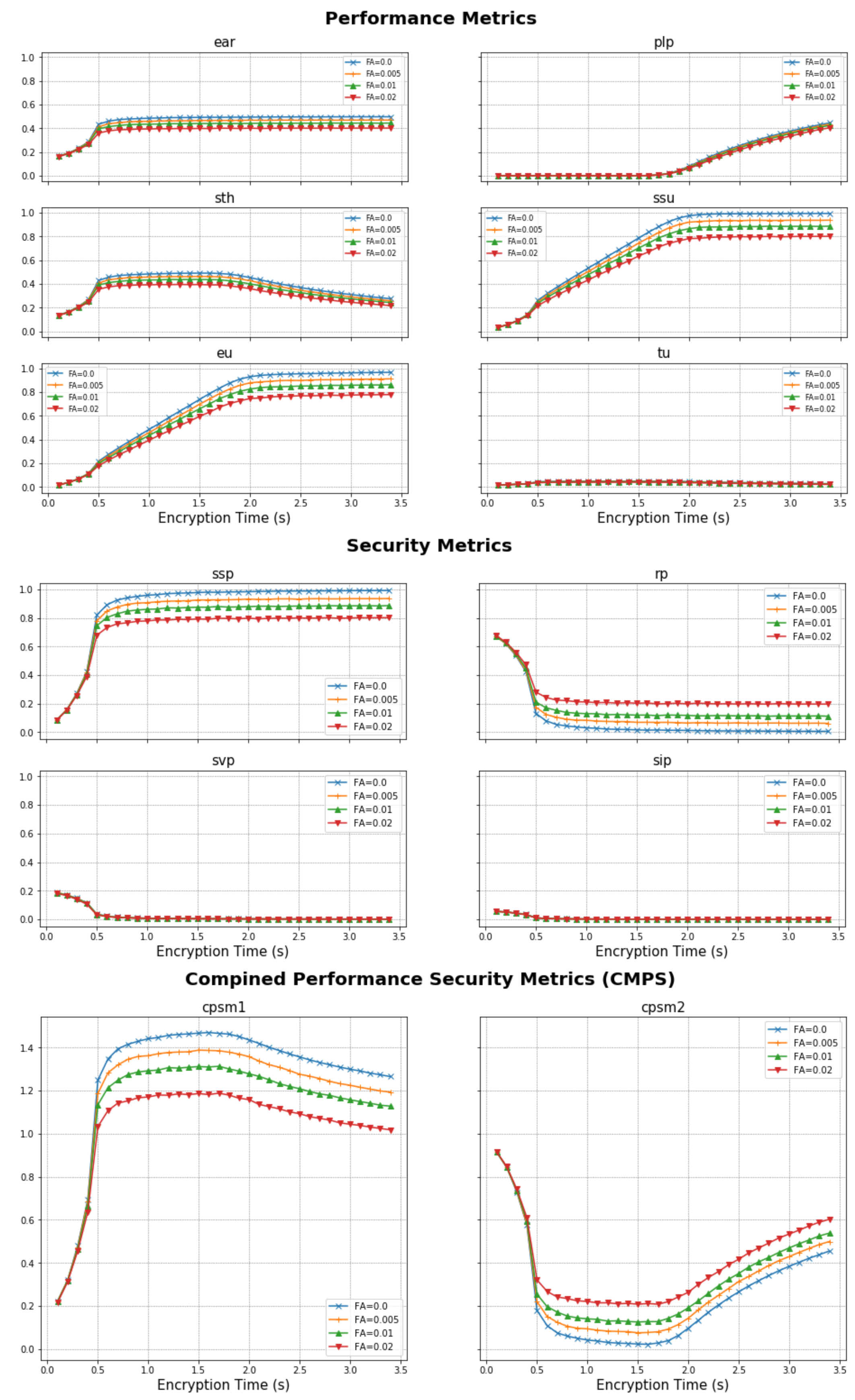

Figure 5.10: Impact of the FA parameter on optimal security keys, showing how different rates of FA (associated with the intrusion detection systems in place) - either FN, FP or both - can possibly affect both performance and security metrics. The Combined metrics reveal that the exact value of the FA parameter can lead into meaningful changes in the optimal keys.

FalseAlarm activities) or fail to detect a real security attack (FN: Case 2 of HIDS, NIDS and VScan activities).

- More specifically, an FP is issued when the system is actually secure but the intrusion inspection erroneously signals an alarm, hence an unrequired recovery. We note that FP rates are associated with increased unavailability, thereby leading to performance implications (reduced throughput). An FN is issued when the system is insecure but the intrusion inspection fails to detect. This can be extremely costly in system life cycle as the attackers have access to the resources without administrators' knowledge. The FN rates thus are associated with increased likelihood of data breaches.

- Figure 5.10 shows that the FA rates have significant impact on both performance and security, as expected; nevertheless, they do not appear to shift (meaningfully) the optimal security key.

5.2.4 Optimal Inspection Interval Scenario

MIIT is the parameter that determines the average inspection interval between two successive executions of a security measure deployed. Thus far, this has been set to a fixed default value of 30 (units of time). In this scenario, however, we let MIIT vary across a range of preselected values so as to allow for sensitivity analysis of performance and security metrics with respect to MIIT.

Table 5.9 below provides settings for all input parameters used in this scenario.

Name	**Value(s)**
BS	[30]
MIAT	[2]
MIIT	[10, 30, 50, 100, 200]
FA	[0.005]
MTT	[0.1]
MIT	[0.1]
MET	[0.1, 0.2, 0.3, ..., 3.3, 3.4]
MTTOV	[12.5, 25, 50, 100, 600, 1100, 1600, 2100, ..., 14600, 15100]
MTTR	[360]
TOEP	[0.5]
VDP	[0.8]

Table 5.9: Input parameters used to identify optimal keys (impact of MIIT parameter)

For each MIIT value given in the table, simulation experiments with respect to security key length are repeated. Figure 5.11 outlines the outcomes for both performance and security metrics as follows.

- Recall that for FA=0, the higher the execution rate (shorter inspection interval), the higher the security of the node overall. However, security inspection tasks are computationally expensive and therefore lower MIITs are likely to impose undue computational overheads on the server, hence more performance loss.

- Non-zero FAs imply that security measures may occasionally fail to identify the true state of security and therefore take wrong actions, either further penalising performance or increasing the chance of data breaches. Here, higher inspection intervals can mean that the node will have to wait for longer periods

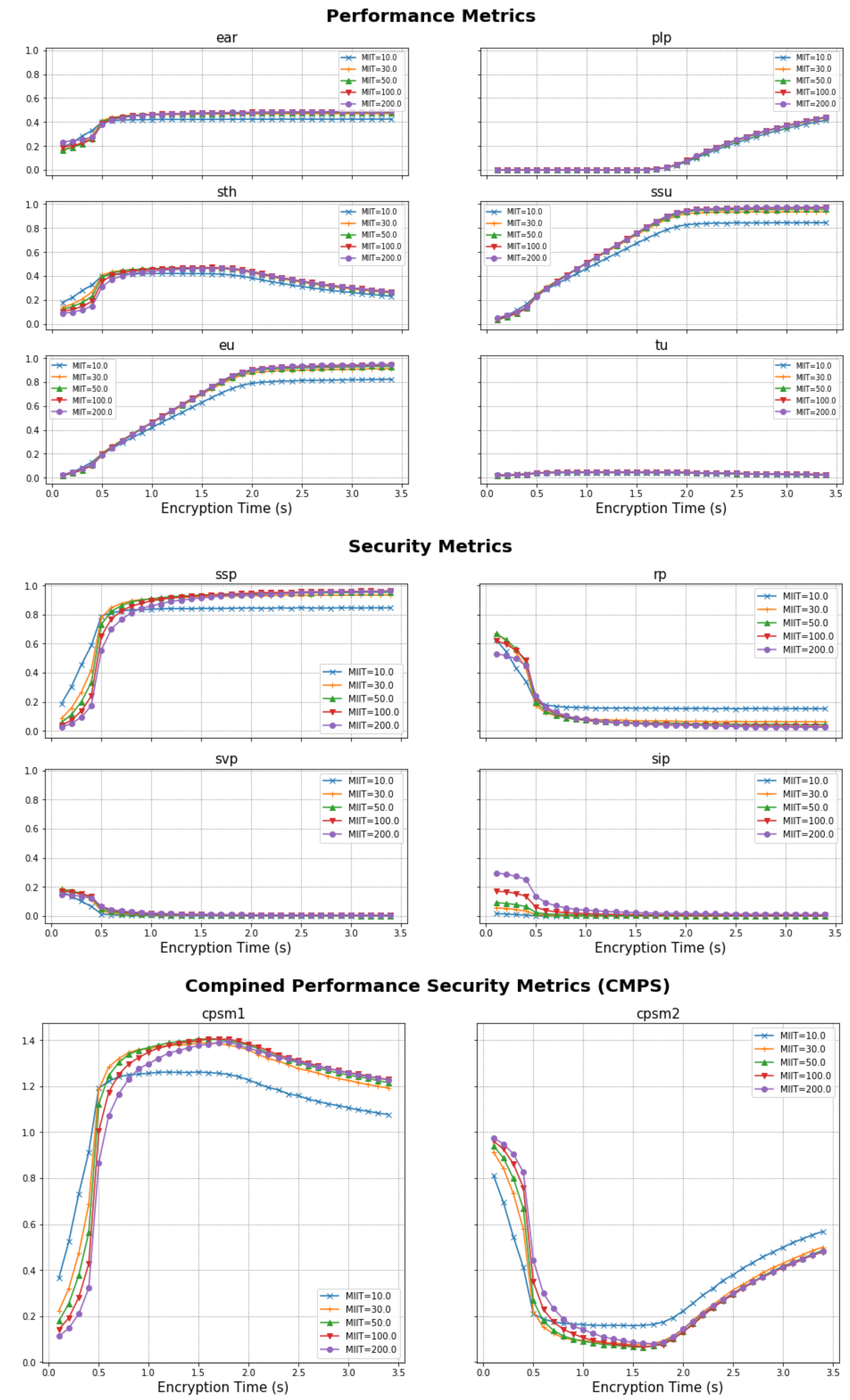

Figure 5.11: Impact of the MIIT parameter on optimal security keys, showing how various settings for the time intervals between security inspections (associated with the intrusion detection systems in place) can possibly affect both performance and security metrics. The Combined metrics reveal that there in fact an optimum interval, not too short not too long, for which the Combined metrics can be further improved and optimised.

(until the following inspections are initiated) once misidentified. In case of lower inspection intervals, these periods can be much shorter; however, due to increased rate of misidentification, the overall times spend in falsely identified security states will increase.

- Both **cpsm1** and **cpsm2** exhibit an explicit optimal value, except for FA=0. This is contrary to common belief that intrusion inspection systems should be operated as often as possible to provide better security assurance. In fact, the results confirm that we need to operate such measures at an optimal rate. In other words, *frequent execution of security measures can be as dangerous and inefficient as operating them infrequently.*

Chapter 6

Conclusions

6.1 Research Summary

Performance and security are closely interrelated such that security measures may require significant amounts of computational resources to execute. Moreover, in the absence of appropriate security measures, frequent security failures are likely to occur, which may potentially lead to catastrophic network performance degradation as well as serious data breaches.

In this thesis, we study optimisation problems for the trade-offs between performance and security as they exist between performance and dependability. While performance metrics are widely studied and well-established, those of security are rarely defined in a strict mathematical sense. We, therefore, conceptualise and formulate security by analogy with dependability so that, like performance, it is also modelled and quantified.

We propose a security state transition diagram which inclusively accommodates for all tangible states an individual computer node can, from security perspective, be found at any given time. We then design and propose, on the basis of the state transition diagram, a stochastic model which allows for quantitative modelling and analysis of combined performance and security.

We adopt a simulation-based modelling approach in order to identify optimal trade-offs between performance and security for a single node of a generic computer network, subject to a number of security threats. Based on a stochastic modelling formalism, we propose a new nodal model which captures more realistically both performance and security aspects, in particular the intertwinements between them. This empirical approach utilises optimal combined metrics of both performance and security and facilitates more sophisticated optimisation studies in this field.

We realise that system parameters can be found that optimise these abstract measures, while they are optimal neither for performance nor for security individually. Based on the proposed simulation modelling framework, credible numerical experiments are carried out, indicating the scope for further work extensions for a systematic performance vs security tuning of computer networks.

6.2 Future Work Directions

In this section, we outline some research areas that may be extended from this thesis as follows.

Energy Consumption

Approaches to extend the system or network lifetime in wireless networks, mostly in wireless sensor networks [3, 17, 31, 33, 59, 81, 102, 116], and MANET [55, 92, 99], have generally been considered in terms of reducing energy consumption. Many energy-efficient algorithms have been devised to prolong network lifetime while meeting performance requirements with minimum energy consumption. A system failure is often defined as when the first node fails [3, 31, 33, 59, 81, 92, 102, 116], or when a majority of nodes (say more than one half) fail due to energy depletion [17].

The focus of this thesis is to model the implications of both security attacks and the legitimate security mechanisms designed to thwart such attacks, and investigate the trade-offs between them. Security attacks, for instance, can take the forms of resource starvation or resource exploitation [76], where attackers make attempts to either access and control unauthorised resources for their own benefits or just simply use them wastefully to minimise the lifetime of a system under attack. In addition, security mechanisms may require significant amount of energy to run throughout the system lifecycle. While both deployment of further security mechanisms and use of longer security keys tend to provide higher security levels, they usually come at the cost of higher energy consumptions; at the same time, the absence of security mechanisms or use of shorter security keys increase the chance of malicious attacks, which in turn can lead to increased wasteful use of energy by intruders. It is clear, therefore, that both attackers and security measures demand substantial amounts of processing power (energy) to accomplish their tasks, one making a wasteful use of the system and the other trying to protect it!

We also realise that, depending on the security status of the system, security measures as well as attacking behaviours may have to dynamically adopt to new conditions; such real world complexities may as well require models with adoptable topologies and dynamically changing service rates in order to accurately represent performance and energy implications of security.

Below are some processes that may consume substantial amount of energy and need to be considered in the models as follows.

- During secure states, extra computational resources are required to run legitimate processes driven by security measures, which are employed for the purpose of security assurance. This is the only cost contribution of security in terms of energy where the system is secure.

- When security state of the system is either vulnerable or insecure, security measures may need to work harder, e.g. due to deviations from normal profiles. They would therefore require even longer service times, thus more energy.

- The system may be subject to various kinds of starvation attacks (CPU, Memory, Resource and Network starvation attacks), implying that the resources will be even more utilised to cope with the demand, now coming from both legitimate and malicious activities. If in any vulnerable or attack states, the extra processing should be reflected in the amount of energy consumption.

Currently, these extra processing opportunities during pre-attack or attack phases are not considered. In future, combined performance and energy models can be introduced taking these into account. In particular, this would be helpful to study systems with strict energy constraints, e.g. in MANETs, where depletion of energy would imply unavailability of the system.

The investigation can possibly aim at identifying the primary sources of energy consumption, their relation with various security threats and trade-offs between energy and security. It is further assumed that computer networks consume more energy when they transition to vulnerable or insecure states. During insecure periods, it is highly likely that adversaries have achieved some success and taken control of the system, at least partially. They would then start executing a variety of tasks to gain unauthorised access to the data (running so many CPU/HD/RAM intensive commands in such a short time for searching, copying, deleting, uploading, etc.) that would not be running otherwise. Even during early stages of security attacks, when attackers probe into the system to learn about potential vulnerabilities, the security measures would probably be busier as they would find some signs of suspicious activities, thereby consuming more energy.

We consider two different cases as follows:

- The computer system assumes no limit in the amount of energy that it can consume, in which case, we may focus on the levels of energy consumption under various security conditions. We realise that,
  - On the one hand, the system is likely to consume more energy where the level of security is very low. This makes sense as such a system would be frequently broken into and, based on the assumptions, the adversaries would intensively utilise the system resources for their own interest.
  - On the other hand, the improved security is very often achieved at the expense of putting a combination of various and lengthier security measures in place. For instance, using longer security keys for (d)encryption which would require longer time to (d)encrypt messages, hence consum-

ing more energy. Also similar simulation experiments can be carried out to examine the impact of such input parameters as FA and MIIT on optimal trade-offs between energy and security.

- We also assume that the computer system has a certain amount of limited energy that, once fully used up, it will stop operating and require a recovery. It is further assumed that the system can be recharged for the same amount of energy every time it goes through recovery. In other words, while the system may fail due to dead battery, it is repairable and, once charged, can start operating again.

Cost Functions

Recall that security mechanisms such as encryption, decryption or security protocols come at a cost in terms of computing resources which may lead to significant performance degradation, e.g. higher response times or lower throughputs. On the other hand, modern computer systems are subject to a variety of malicious attacks which, in the absence of security mechanisms, can easily end up in long unavailable conditions or data leakage to unauthorised users. More accurate cost factors would depend on the length of the time such system resources are exposed to intruders and the value and volume of unauthorised data accessed, just to name a few. The level of increased energy consumption can also be considered as a cost factor which is caused by both security threats and security measures in place.

Wolter and Reinecke [112] provide an in-depth discussion of security cost based on two simplified gain and lost functions, where the authors formulate an optimisation problem for encryption. That is, the longer the security key, the longer the encryption time, the higher the cost. At the same time, a security incident has associated costs as well. Therefore, the security key must have an optimal length so as to keep encryption costs and the cost of security related incidents together as low as possible. Vice versa, normal operation of a system achieves revenue which is reduced by encryption cost and, at a much higher degree, by the recovery cost from a security incident. In order to maximise revenue, the encryption key is chosen such that the encryption cost is reasonably low and security incidents occur only very rarely.

There are a number of areas where these cost functions can be improved both in terms of accuracy and practicality. Having employed SAN paradigm, we believe that more detailed rate rewards and reward functions can be devised to provide a more realistic reflection of cost and implications of security breaches. This may include

- The amount and value of the data exposed to potential intruders
- How long did they have access to such data
- An estimation of system unavailability period
- The amount of data purged which would need reprocessing
- Implications of performance degradation such as longer waiting and response times, buffer lengths
- Increased packet loss probabilities

Other Considerations

In addition to Energy Consumption and Cost Functions, the current research can be extended to generalise and incorporate more features with respect to some other aspects such as:

- Modelling a Network: The immediate extension could be a network of several hosts. This would allow for study of similar performance and security concepts and measures across a typical computer network. It would also enable investigation into scalability issues with regard to security protocols involved.
- Alternative Time Distributions: Lessons learned may be taken into consideration to represent maturer detection systems [52] and more sophisticated attack scenarios, providing more accurate and reliable predictive outcomes. Cho [18] consider three attacker functions: Logarithmic, Linear and Polynomial time attackers, as well as three similar detection functions. Others propose even a

longer list of distributions sensible in the context of security analysis including deterministic, exponential, hyper-exponential, hypo-exponential, weibull, gamma and log-logistic [60, 63].

- Modelling of Various Security Threats/Measures: For the purpose of this thesis, we only vary the encryption times (security key lengths) and investigate the impacts on performance and security aspects of the node. Although performance implications of other security mechanisms such as anti-virus programs may be easily understood and modelled, less is known about how security levels quantitatively would be affected with respect to different techniques/algorithms involved in such measures. Therefore, quantification of other security measures in a similar way to encryption key lengths will allow for examination of real life systems where a number of different security threats and their respective measures can be actively operating at the same time.

- Cooperative Detection and Attack: Our security modules are currently modelled so that both the attack submodels and their respective detection submodels operate independently and in a self-initiated manner, except for the times where a successful attack of either type forces the whole system to freeze. This can be extended such that IDRSs are both distributed and cooperative [115], thereby representing a more realistic model with improved accuracy [54]. The models can equally be extended to reflect the coordinated attacks targeting the system simultaneously from various angles.

- Simultaneous Attacks: We realise that, in a real-life, there is no way to limit the number of attackers who are simultaneously targeting a system. In other words, while it is sensible to allow for simultaneous attacks across different attack types[1], it should perfectly be fine to assume that there might be more than one for each of these attack types at any given moment who are making attempts to intrude the system.

[1]As is the case in this research thesis where attackers of insiders, outsiders and viruses can all be targeting the node independent from one another and possibly at the same time

- Recall that the SSTD models are assumed to work independently. However, it does not always make sense to assume a vulnerability is known to a certain group of attackers but not to the others. In a real-life, if a vulnerability is disclosed, one should expect all types of attackers, soon after, to become aware of it, hence a dependency among SSTD models!

- Performance Implications of Security Attacks: In addition to security measures, some attack types may be computationally intensive too. The DDoS is probably the most common attack of this type, which is usually accomplished by flooding the system with superfluous requests in an attempt to overload it and prevent some or all legitimate requests from being fulfilled. In its less destructive form, an attacker may even attack the target system in order to solely take advantage of its resources without intending to cause any harm at all. While such performance implications of the attacks have not been considered in this thesis, they appear really intriguing.

- Real Life Data: Any data from real life security attack/detect scenarios would extremely be helpful in refinement of the models and the parameters involved. Since such data is hardly made available by companies, we aim to design and deploy a pilot Local Area Network (LAN) which is subject to various security threats and has respective security measures to tackle them. Data collected this way should provide useful insights into modelling and parameterisation processes discussed earlier in this thesis.

Bibliography

[1] Mohamed Abdlhamed, Kashif Kifayat, Qi Shi, and William Hurst. “A System for Intrusion Prediction in Cloud Computing”. In: *Proceedings of the International Conference on Internet of Things and Cloud Computing*. ICC ’16. Cambridge, United Kingdom: ACM, 2016, 35:1–35:9. ISBN: 978-1-4503-4063-2. DOI: `10.1145/2896387.2896420`. URL: `http://doi.acm.org.brad.idm.oclc.org/10.1145/2896387.2896420`.

[2] Mohamed Abdlhamed, Kashif Kifayat, Qi Shi, and William Hurst. “Intrusion Prediction Systems”. In: *Information Fusion for Cyber-Security Analytics*. Ed. by Izzat M Alsmadi, George Karabatis, and Ahmed Aleroud. Cham: Springer International Publishing, 2017, pp. 155–174. ISBN: 978-3-319-44257-0. DOI: `10.1007/978-3-319-44257-0_7`. URL: `https://doi.org/10.1007/978-3-319-44257-0_7`.

[3] A. Alfieri, A. Bianco, P. Brandimarte, and C.F. Chiasserini. “Maximizing system lifetime in wireless sensor networks”. In: *European Journal of Operational Research* 181.1 (2007), pp. 390–402. ISSN: 0377-2217. DOI: `https://doi.org/10.1016/j.ejor.2006.05.037`. URL: `http://www.sciencedirect.com/science/article/pii/S0377221706004255`.

[4] Arnold O. Allen. *Probability, Statistics, and Queueing Theory with Computer Science Applications*. San Diego, CA, USA: Academic Press Professional, Inc., 1990. ISBN: 0-12-051051-0.

[5] Jaafar Almasizadeh and Mohammad Abdollahi Azgomi. “A stochastic model of attack process for the evaluation of security metrics”. In: *Computer Networks* 57.10 (2013). Towards a Science of Cyber Security Security and Identity Architecture for the Future Internet, pp. 2159–2180. ISSN: 1389-1286. DOI: `http://dx.doi.org/10.1016/j.comnet.2013.03.011`. URL: `http://www.sciencedirect.com/science/article/pii/S1389128613000856`.

[6] William A Arbaugh, William L Fithen, and John McHugh. “Windows of vulnerability: A case study analysis”. In: *Computer* 33.12 (2000), pp. 52–59.

[7] Florian Arnold, Holger Hermanns, Reza Pulungan, and Mariëlle Stoelinga. “Time-Dependent Analysis of Attacks.” In: *POST* 14 (2014), pp. 285–305.

[8] W. Arnold and G. Tesauro. “Automatically Generated WIN32 Heuristic Virus Detection”. In: *Proceedings of the Virus Bulletin Conference, Orlando, Florida, USA, Virus Bulletin Ltd.* 2000.

[9] Algirdas Avizienis, J-C Laprie, Brian Randell, and Carl Landwehr. “Basic concepts and taxonomy of dependable and secure computing”. In: *IEEE transactions on dependable and secure computing* 1.1 (2004), pp. 11–33.

[10] J. Banks, J. S. Carson, B. L. Nelson, and D. Nicol. *Discrete-Event System Simulation*. 5th ed. Prentice Hall, 2010. ISBN: 0136062121.

[11] Dimitri P Bertsekas, Robert G Gallager, and Pierre Humblet. *Data networks.* Vol. 2. Prentice-hall Englewood Cliffs, NJ, 1987.

[12] Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor S Trivedi. *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications.* John Wiley & Sons, 2006.

[13] Brian Bowen, Malek Ben Salem, Shlomo Hershkop, Angelos Keromytis, and Salvatore Stolfo. "Designing host and network sensors to mitigate the insider threat". In: *IEEE Security & Privacy* 7.6 (2009), pp. 22–29.

[14] Sarah Brocklehurst, Bev Littlewood, Tomas Olovsson, and Erland Jonsson. "On measurement of operational security [software reliability]". In: *Computer Assurance, 1994. COMPASS'94 Safety, Reliability, Fault Tolerance, Concurrency and Real Time, Security. Proceedings of the Ninth Annual Conference on.* IEEE, pp. 257–266.

[15] US-CERT. *Virsu Basics.* 2017. URL: `https://www.us-cert.gov/publications/virus-basics`.

[16] I-R Chen, Jin-Hee Cho, and Ding-Chau Wang. "Performance characteristics of region-based group key management in mobile ad hoc networks". In: *Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006. IEEE International Conference on.* Vol. 1. IEEE. 2006, 8–pp.

[17] Yunxia Chen and Qing Zhao. "On the lifetime of wireless sensor networks". In: *IEEE Communications Letters* 9.11 (Nov. 2005), pp. 976–978. ISSN: 1089-7798. DOI: `10.1109/LCOMM.2005.11010`.

[18] Jin-Hee Cho. "Design and analysis of QoS-aware key management and intrusion detection protocols for secure mobile group communications in wireless networks". PhD thesis. Virginia Tech, 2008.

[19] Jin-Hee Cho and Ray Chen. "On design tradeoffs between security and performance in wireless group communicating systems". In: *Secure Network Protocols, 2005.(NPSec). 1st IEEE ICNP Workshop on.* IEEE. 2005, pp. 13–18.

[20] Jin-Hee Cho and Ray Chen. "Performance analysis of distributed intrusion detection protocols for mobile group communication systems". In: *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on.* IEEE. 2009, pp. 1–8.

[21] Jin-Hee Cho and Ray Chen. "Performance analysis of hierarchical group key management integrated with adaptive intrusion detection in mobile ad hoc networks". In: *Performance Evaluation* 68.1 (2011), pp. 58–75.

[22] Jin-Hee Cho, Ray Chen, and Mohamed Eltoweissy. "On optimal batch rekeying for secure group communications in wireless networks". In: *Wireless Networks* 14.6 (2008), pp. 915–927.

[23] Jin-Hee Cho, Ray Chen, and Mohamed Eltoweissy. "Optimization of batch rekey interval for secure group communications in wireless networks". In: *Wireless Networks, Communications and Mobile Computing, 2005 International Conference on.* Vol. 1. IEEE. 2005, pp. 522–527.

[24] Jin-Hee Cho, Ray Chen, and Phu-Gui Feng. "Effect of intrusion detection on reliability of mission-oriented mobile group systems in mobile ad hoc networks". In: *IEEE Transactions on Reliability* 59.1 (2010), pp. 231–241.

[25] Jin-Hee Cho, Ray Chen, and Phu-Gui Feng. “Performance Analysis of Dynamic Group Communication Systems with Intrusion Detection Integrated with Batch Rekeying in Mobile Ad Hoc Networks”. In: *22nd International Conference on Advanced Information Networking and Applications-Workshops*. 2008.

[26] CISCO. *What Is the Difference: Viruses, Worms, Trojans, and Bots?* Dec. 2017. URL: `http://www.cisco.com/c/en/us/about/security-center/virus-differences.html`.

[27] *Cisco: What Is Network Security.* `https://www.cisco.com/c/en/us/products/security/what-is-network-security.html/`. Accessed: 2017-11-07.

[28] CIS-SM10. *The CIS Security Metrics v1.1.0.* 2010.

[29] Graham Clark, Tod Courtney, David Daly, Dan Deavours, Salem Derisavi, Jay M Doyle, William H Sanders, and Patrick Webster. “The Mobius modeling tool”. In: *Petri Nets and Performance Models, 2001. Proceedings. 9th International Workshop on.* IEEE. 2001, pp. 241–250.

[30] Hervé Debar, Marc Dacier, and Andreas Wespi. “Towards a taxonomy of intrusion-detection systems”. In: *Computer Networks* 31.8 (1999), pp. 805–822.

[31] E. J. Duarte-Melo and Mingyan Liu. “Analysis of energy consumption and lifetime of heterogeneous wireless sensor networks”. In: *Global Telecommunications Conference, 2002. GLOBECOM ’02. IEEE.* Vol. 1. Nov. 2002, 21–25 vol.1. DOI: `10.1109/GLOCOM.2002.1188034`.

[32] Diaa Salama Abd Elminaam, Hatem Mohamed Abdual-Kader, and Mohiy Mohamed Hadhoud. “Evaluating the performance of symmetric encryption algorithms.” In: *IJ Network Security* 10.3 (2010), pp. 216–222.

[33] Moez Esseghir and Nizar Bouabdallah. “Node density control for maximizing wireless sensor network lifetime”. In: *Int. Journal of Network Management* 18 (2008), pp. 159–170.

[34] Reiner Funke, Andreas Grote, and Hans-Ulrich Heiss. “Performance evaluation of firewalls in gigabit-networks”. In: *Proceedings of the Symposium on Performance Evaluation of Computer and Telecommunication Systems*. Citeseer. 1999.

[35] Katerina Goševa-Popstojanova, Feiyi Wang, Rong Wang, Fengmin Gong, Kalyanaraman Vaidyanathan, Kishor Trivedi, and Balamurugan Muthusamy. “Characterizing intrusion tolerant systems using a state transition model”. In: *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX’01. Proceedings.* Vol. 2. IEEE. 2001, pp. 211–221.

[36] Christopher Griffin, Bharat Madan, and T Trivedi. “State space approach to security quantification”. In: *Computer Software and Applications Conference, 2005. COMPSAC 2005. 29th Annual International.* Vol. 2. IEEE. 2005, pp. 83–88.

[37] Mark D Hill. “What is scalability?” In: *ACM SIGARCH Computer Architecture News* 18.4 (1990), pp. 18–21.

[38] Sarah J Hood and Peter D Welch. “Experimental design issues in simulation with examples from semiconductor manufacturing”. In: *Proceedings of the 24th conference on Winter simulation.* ACM. 1992, pp. 255–263.

[39] Sarah J Hood and Peter D Welch. “Response surface methodology and its application in simulation”. In: *Proceedings of the 25th conference on Winter simulation*. ACM. 1993, pp. 115–122.

[40] Raj Jain. *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling*. Wiley Professional Computing. Wiley, 1991, pp. I–XXVII, 1–685. ISBN: 978-0-471-50336-1.

[41] Erland Jonsson and Mikael Andersson. “On the quantitative assessment of behavioural security”. In: *Information Security and Privacy*. Springer. 1996, pp. 228–241.

[42] Erland Jonsson and Tomas Olovsson. “A quantitative model of the security intrusion process based on attacker behavior”. In: *IEEE Transactions on Software Engineering* 23.4 (1997), pp. 235–245.

[43] Erland Jonsson and Tomas Olovsson. “On the integration of security and dependability in computer systems”. In: *IASTED International Conference on Reliability, Quality Control and Risk Assessment Washington DC, USA, 1992, ISBN 0-88986-171-4*. 1992, pp. 93–97.

[44] Chris Karlof and David Wagner. “Secure routing in wireless sensor networks: Attacks and countermeasures”. In: *Ad hoc networks* 1.2 (2003), pp. 293–315.

[45] Tom Karygiannis and Les Owens. “Wireless network security”. In: *NIST special publication* 800 (2002), p. 48.

[46] W. David Kelton. “Designing Simulation Experiments”. In: *Proceedings of the 31st Conference on Winter Simulation: Simulation—a Bridge to the Future - Volume 1*. WSC ’99. Phoenix, Arizona, USA: ACM, 1999, pp. 33–38. ISBN: 0-7803-5780-9. DOI: 10.1145/324138.324146. URL: http://doi.acm.org.brad.idm.oclc.org/10.1145/324138.324146.

[47] Kimberly Kiefer. *Information Security: A Legal, Business, and Technical Handbook*. ABA Publishing, 2004. ISBN: 1590313003.

[48] Jack PC Kleijnen. “Experimental design for sensitivity analysis, optimization, and validation of simulation models”. In: *Handbook of simulation* 1998 (1998), pp. 173–223.

[49] Demetres D. Kouvatsos and Guzlan M. A. Miskeen. “Performance Related Security Modelling and Evaluation of RANETs”. In: *Wireless Personal Communications* 64.3 (2012), pp. 523–546. ISSN: 1572-834X. DOI: 10.1007/s11277-012-0599-1. URL: http://dx.doi.org/10.1007/s11277-012-0599-1.

[50] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach (6th Edition)*. 6th. Pearson, 2012. ISBN: 0132856204, 9780132856201.

[51] C Lamprecht, A Van Moorsel, P Tomlinson, and N Thomas. “Investigating the efficiency of cryptographic algorithms in online transactions”. In: *International Journal of Simulation: Systems, Science & Technology* 7.2 (2006), pp. 63–75.

[52] James LaPiedra. *The Information Security Process Prevention, Detection and Response*. Tech. rep. SANS Institute, 2002.

[53] Averill M Law, W David Kelton, and W David Kelton. *Simulation modeling and analysis*. Vol. 3. McGraw-Hill New York, 2007.

[54] Wenke Lee, Wei Fan, Matthew Miller, Salvatore J Stolfo, and Erez Zadok. "Toward cost-sensitive modeling for intrusion detection and response". In: *Journal of computer security* 10.1-2 (2002), pp. 5–22.

[55] Zongpeng Li and Baochun Li. "Probabilistic Power Management for Wireless Ad Hoc Networks". In: *Mobile Networks and Applications* 10.5 (2005), pp. 771–782. ISSN: 1572-8153. DOI: 10.1007/s11036-005-3370-y. URL: http://dx.doi.org/10.1007/s11036-005-3370-y.

[56] Bev Littlewood, Sarah Brocklehurst, Norman Fenton, Peter Mellor, Stella Page, David Wright, John Dobson, John McDermid, and Dieter Gollmann. "Towards operational measures of computer security". In: *Journal of Computer Security* 2.2-3 (1993), pp. 211–229.

[57] John Lowry. "An initial foray into understanding adversary planning and courses of action". In: *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX'01. Proceedings.* Vol. 1. IEEE. 2001, pp. 123–133.

[58] Edward A Luke. "Defining and measuring scalability". In: *Scalable Parallel Libraries Conference, 1993., Proceedings of the.* IEEE. 1993, pp. 183–186.

[59] Yong Ma and J. H. Aylor. "System lifetime optimization for heterogeneous sensor networks with a hub-spoke technology". In: *IEEE Transactions on Mobile Computing* 3.3 (July 2004), pp. 286–294. ISSN: 1536-1233. DOI: 10.1109/TMC.2004.27.

[60] Bharat B Madan, K Goševa-Popstojanova, Kalyanaraman Vaidyanathan, and Kishor S Trivedi. "Modeling and quantification of security attributes of software systems". In: *Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on.* IEEE. 2002, pp. 505–514.

[61] Bharat B Madan, Katerina Goševa-Popstojanova, Kalyanaraman Vaidyanathan, and Kishor S Trivedi. "A method for modeling and quantifying the security attributes of intrusion tolerant systems". In: *Performance Evaluation* 56.1 (2004), pp. 167–186.

[62] Marco Ajmone Marsan, Gianfranco Balbo, Gianni Conte, Susanna Donatelli, and Giuliana Franceschinis. *Modelling with generalized stochastic Petri nets.* John Wiley & Sons, Inc., 1994.

[63] John I McCool. *Probability and Statistics With Reliability, Queuing and Computer Science Applications.* 2003.

[64] D. A. Menascé. "Security performance". In: *IEEE Internet Computing* 7.3 (May 2003), pp. 84–87. ISSN: 1089-7801. DOI: 10.1109/MIC.2003.1200305.

[65] D.A. Menascé, V.A.F. Almeida, L.W. Dowdy, and L. Dowdy. *Performance by Design: Computer Capacity Planning by Example.* Prentice Hall PTR, 2004. ISBN: 9780130906731. URL: https://books.google.co.uk/books?id=sxE931Medm4C.

[66] Tianhui Meng. "Security and Performance Tradeoff Analysis of Offloading Policies in Mobile Cloud Computing". PhD thesis. Freie Universität Berlin, 2017.

[67] Tianhui Meng, Qiushi Wang, and Katinka Wolter. "Model-Based Quantitative Security Analysis of Mobile Offloading Systems Under Timing Attacks". In: *Analytical and Stochastic Modelling Techniques and Applications: 22nd International Conference, ASMTA 2015, Albena, Bulgaria, May 26-29, 2015. Proceedings*. Ed. by Marco Gribaudo, Daniele Manini, and Anne Remke. Cham: Springer International Publishing, 2015, pp. 143–157. ISBN: 978-3-319-18579-8. DOI: `10.1007/978-3-319-18579-8_11`. URL: `https://doi.org/10.1007/978-3-319-18579-8_11`.

[68] John F. Meyer et al. "On evaluating the performability of degradable computing systems". In: *IEEE Transactions on computers* 29.8 (1980), pp. 720–731.

[69] John F. Meyer. "Performability modeling: Back to the future?" In: *Performability Modeling of Computer and Communication Systems, 2007 Proceedings, The 8th International Workshop on Workshop*. Vol. 1. CTIT. 2007, pp. 5–9.

[70] Guzlan M. A. Miskeen, Demetres D. Kouvatsos, and Esmaeil H. Zadeh. "An Exposition of Performance-Security Trade-offs in RANETs Based on Quantitative Network Models". In: *Wireless Personal Communications* 70.3 (2013), pp. 1121–1146. ISSN: 1572-834X. DOI: `10.1007/s11277-013-1105-0`. URL: `http://dx.doi.org/10.1007/s11277-013-1105-0`.

[71] Guzlan Mohamed Alzaroug Miskeen. "Performance and Security Trade-offs in High-Speed Networks. An investigation into the performance and security modelling and evaluation of high-speed networks based on the quantitative analysis and experimentation of queueing networks and generalised stochastic Petri nets." PhD thesis. University of Bradford, 2014.

[72] Leonardo Montecchi, Nicola Nostro, Andrea Ceccarelli, Giuseppe Vella, Antonio Caruso, and Andrea Bondavalli. "Model-based Evaluation of Scalability and Security Tradeoffs: a Case Study on a Multi-Service Platform". In: *Electronic Notes in Theoretical Computer Science* 310 (2015), pp. 113–133. ISSN: 1571-0661. DOI: `http://dx.doi.org/10.1016/j.entcs.2014.12.015`. URL: `http://www.sciencedirect.com/science/article/pii/S1571066114000991`.

[73] Joel A Nachlas. *Reliability engineering: probabilistic models and maintenance methods*. CRC Press, 2017.

[74] Derek L. Nazareth and Jae Choi. "A system dynamics model for information security management". In: *Information & Management* 52.1 (2015), pp. 123–134. ISSN: 0378-7206. DOI: `https://doi.org/10.1016/j.im.2014.10.009`. URL: `http://www.sciencedirect.com/science/article/pii/S0378720614001335`.

[75] David M Nicol, William H Sanders, and Kishor S Trivedi. "Model-based evaluation: from dependability to security". In: *IEEE Transactions on dependable and secure computing* 1.1 (2004), pp. 48–65.

[76] David L Olson and Desheng Wu. *Enterprise risk management models*. Springer, 2010.

[77] Thomas Parsons. *A False Positive Prevention Framework for Non-Heuristic Anti-Virus Signatures*. Tech. rep. Symantec, Security Response, 2009.

[78] Marcus Pendleton, Richard Garcia-Lebron, Jin-Hee Cho, and Shouhuai Xu. “A survey on systems security metrics”. In: *ACM Computing Surveys (CSUR)* 49.4 (2016), p. 62.

[79] PEPA. *Performance Evaluation Process Algebra*. Dec. 2017. URL: `http://www.dcs.ed.ac.uk/pepa/`.

[80] D. M. Perry. “THE (LIFE AND) DEATH OF THE PATTERN FILE”. In: *Proceedings of The 18th Virus Bulletin International Conference, Ottawa, Canada, Virus Bulletin Ltd.* 2008.

[81] S. Pollin, R. Mangharam, B. Bougard, L. Van Der Perre, I. Moerman, R. Rajkumar, and F. Catthoor. “MEERA: Cross-Layer Methodology for Energy Efficient Resource Allocation in Wireless Networks”. In: *IEEE Transactions on Wireless Communications* 7.1 (Jan. 2008), pp. 98–109. ISSN: 1536-1276. DOI: `10.1109/TWC.2008.05356`.

[82] Jim Reavis. *The Ongoing Malware Threat: How Malware Infects Websites and Harms Businesses and What You Can Do to Stop It.* Tech. rep. Symantec Norton, 2012.

[83] Eliot Rich, Ignacio J Martinez-Moyano, Stephen Conrad, Dawn M Cappelli, Andrew P Moore, Timothy J Shimeall, David F Andersen, Jose J Gonzalez, Robert J Ellison, Howard F Lipson, et al. “Simulating insider cyber-threat risks: A model-based case and a case-based model”. In: *Proceedings of the 23rd International Conference of the System dynamics Society*. The System Dynamics Society. 2005, pp. 17–21.

[84] M. Riek, R. Bohme, and T. Moore. “Measuring the Influence of Perceived Cybercrime Risk on Online Service Avoidance”. In: *IEEE Transactions on Dependable and Secure Computing* 13.2 (Mar. 2016), pp. 261–273. ISSN: 1545-5971. DOI: `10.1109/TDSC.2015.2410795`.

[85] M.N.O. Sadiku and S.M. Musa. *Performance Analysis of Computer Networks*. Springer International Publishing, 2013. ISBN: 9783319016450. URL: `https://books.google.co.uk/books?id=LulTngEACAAJ`.

[86] Robin A Sahner, Kishor Trivedi, and Antonio Puliafito. *Performance and reliability analysis of computer systems: an example-based approach using the SHARPE software package*. Springer Science & Business Media, 2012.

[87] Khaled Salah, Khalid Elbadawi, and Raouf Boutaba. “Performance modeling and analysis of network firewalls”. In: *IEEE Transactions on network and service management* 9.1 (2012), pp. 12–21.

[88] W. H. Sanders and the Board of Trustees of the University of Illinois. *The Mobius Tool*. 2017. URL: `https://www.mobius.illinois.edu/`.

[89] William H Sanders, Tod Courtney, Daniel Deavours, David Daly, Salem Derisavi, and Vinh Lam. *Multi-formalism and multi-solution-method modeling frameworks: The Möbius approach*. na, 2003.

[90] William H Sanders and John F Meyer. “Stochastic activity networks: Formal definitions and concepts”. In: *Lectures on Formal Methods and Performance-Analysis*. Springer, 2001, pp. 315–343.

[91] Alireza Shameli Sendi, Michel Dagenais, Masoume Jabbarifar, and Mario Couture. “Real Time Intrusion Prediction based on Optimized Alerts with Hidden Markov Model.” In: *JNW* 7.2 (2012), pp. 311–321.

[92] Cigdem Sengul and Robin H. Kravets. "TITAN: on-demand topology management in ad hoc networks". In: *SIGMOBILE Mob. Comput. Commun. Rev.* 9.1 (2005), 77–82. ISSN: 1559-1662. DOI: http://doi.acm.org/10.1145/1055959.1055972.

[93] Chirag Sheth and Rajesh Thakker. "Performance evaluation and comparison of network firewalls under DDoS attack". In: *International Journal of Computer Network and Information Security* 5.12 (2013), p. 60.

[94] George Silowash, Dawn Cappelli, Andrew Moore, Randall Trzeciak, Timothy J Shimeall, and Lori Flynn. *Common sense guide to mitigating insider threats 4th edition.* Tech. rep. DTIC Document, 2012.

[95] Nidhi Singhal and JPS Raina. "Comparative analysis of AES and RC4 algorithms for better utilization". In: *International Journal of Computer Trends and Technology* 2.6 (2011), pp. 177–181.

[96] George Anastasios Spanos and Tracy Bradley Maples. "Performance study of a selective encryption scheme for the security of networked, real-time video". In: *Computer Communications and Networks, 1995. Proceedings., Fourth International Conference on.* IEEE. 1995, pp. 2–10.

[97] Natalia Stakhanova, Samik Basu, and Johnny Wong. "A taxonomy of intrusion response systems". In: *International Journal of Information and Computer Security* 1.1-2 (2007), pp. 169–184.

[98] William Stallings and Mohit P Tahiliani. *Cryptography and network security: principles and practice.* Vol. 6. Pearson London, 2014.

[99] Michael Steiner, Gene Tsudik, and Michael Waidner. "Diffie-Hellman Key Distribution Extended to Group Communication". In: *Proceedings of the 3rd ACM Conference on Computer and Communications Security.* CCS '96. New Delhi, India: ACM, 1996, pp. 31–37. ISBN: 0-89791-829-0. DOI: 10.1145/238168.238182. URL: http://doi.acm.org/10.1145/238168.238182.

[100] Aaron James Stillman. "Model composition within the Mobius modeling framework". MA thesis. Citeseer, 1999.

[101] Symantec. *Understanding heuristics: Symantec's bloodhound technology.* Tech. rep. Symantec White Paper Series, Volume XXXIV, 1997.

[102] Chiping Tang and P. K. McKinley. "Energy Optimization under Informed Mobility". In: *IEEE Transactions on Parallel and Distributed Systems* 17.9 (Sept. 2006), pp. 947–962. ISSN: 1045-9219. DOI: 10.1109/TPDS.2006.122.

[103] James Tarala. *Reducing Risk Through Prevention.* Tech. rep. SANS Institute, 2013.

[104] Jawahar Thakur and Nagesh Kumar. "DES, AES and Blowfish: Symmetric key cryptography algorithms simulation based performance analysis". In: *International journal of emerging technology and advanced engineering* 1.2 (2011), pp. 6–12.

[105] Nigel Thomas. "Performability of a Secure Electronic Voting Algorithm". In: *Electronic Notes in Theoretical Computer Science* 128.4 (2005). Proceedings of the First International Workshop on Practical Applications of Stochastic Modelling (PASM 2004), pp. 45–58. ISSN: 1571-0661. DOI: http://dx.doi.org/10.1016/j.entcs.2005.01.012. URL: http://www.sciencedirect.com/science/article/pii/S1571066105001878.

[106] Kishor S Trivedi, Dong Seong Kim, Arpan Roy, and Deep Medhi. "Dependability and security models". In: *Design of Reliable Communication Networks, 2009. DRCN 2009. 7th International Workshop on.* IEEE. 2009, pp. 11–20.

[107] Derek Uluski, Micha Moffie, and David Kaeli. "Characterizing antivirus workload execution". In: *ACM SIGARCH Computer Architecture News* 33.1 (2005), pp. 90–98.

[108] Vilhelm Verendel. "Quantified Security is a Weak Hypothesis: A Critical Survey of Results and Assumptions". In: *Proceedings of the 2009 Workshop on New Security Paradigms Workshop.* NSPW '09. Oxford, United Kingdom: ACM, 2009, pp. 37–50. ISBN: 978-1-60558-845-2. DOI: 10.1145/1719030.1719036. URL: http://doi.acm.org/10.1145/1719030.1719036.

[109] Dazhi Wang, Bharat B. Madan, and Kishor S. Trivedi. "Security Analysis of SITAR Intrusion Tolerance System". In: *Proceedings of the 2003 ACM Workshop on Survivable and Self-regenerative Systems: In Association with 10th ACM Conference on Computer and Communications Security.* SSRS '03. Fairfax, VA: ACM, 2003, pp. 23–32. ISBN: 1-58113-784-2. DOI: 10.1145/1036921.1036924. URL: http://doi.acm.org/10.1145/1036921.1036924.

[110] Yang Wang, Chuang Lin, and Quan-Lin Li. "Performance analysis of email systems under three types of attacks". In: *Performance Evaluation* 67.6 (2010), pp. 485–499. ISSN: 0166-5316. DOI: http://dx.doi.org/10.1016/j.peva.2010.01.003. URL: http://www.sciencedirect.com/science/article/pii/S0166531610000040.

[111] M. M. Williamson. "Using behaviour to detect and classify information-stealing malware". In: *Proceedings of the 15th Virus Bulletin International Conference, Dublin, Ireland, Virus Bulletin Ltd.* 2005.

[112] Katinka Wolter and Philipp Reinecke. "Performance and Security Tradeoff". In: *Proceedings of the Formal Methods for Quantitative Aspects of Programming Languages, and 10th International Conference on School on Formal Methods for the Design of Computer, Communication and Software Systems.* SFM'10. Bertinoro, Italy: Springer-Verlag, 2010, pp. 135–167. ISBN: 3-642-13677-X, 978-3-642-13677-1. URL: http://dl.acm.org/citation.cfm?id=1875868.1875872.

[113] Junjie Xu and Wenhui Su. *Performance Evaluations of Cisco ASA and Linux IPTables Firewall Solutions.* 2013.

[114] Yongguang Zhang and Wenke Lee. "Intrusion detection in wireless ad-hoc networks". In: *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking.* ACM. 2000, pp. 275–283.

[115] Yongguang Zhang, Wenke Lee, and Yi-An Huang. "Intrusion detection techniques for mobile wireless networks". In: *Wireless Networks* 9.5 (2003), pp. 545–556.

[116] Qing Zhao and L. Tong. "Energy efficiency of large-scale wireless networks: proactive versus reactive networking". In: *IEEE Journal on Selected Areas in Communications* 23.5 (May 2005), pp. 1100–1112. ISSN: 0733-8716. DOI: 10.1109/JSAC.2005.845411.

[117] Yishi Zhao and Nigel Thomas. "Efficient solutions of a PEPA model of a key distribution centre". In: *Performance Evaluation* 67.8 (2010), pp. 740–756.

[118] Yishi Zhao and Nigel Thomas. *Modelling secure secret key exchange using stochastic process algebra*. University of Newcastle upon Tyne-Computing Science, 2007.

[119] Vasilios Zorkadis. “Security versus performance requirements in data communication systems”. In: *Computer Security — ESORICS 94: Third European Symposium on Research in Computer Security Brighton, United Kingdom, November 7–9, 1994 Proceedings*. Ed. by Dieter Gollmann. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 19–30. ISBN: 978-3-540-49034-0. DOI: `10.1007/3-540-58618-0_54`. URL: `http://dx.doi.org/10.1007/3-540-58618-0_54`.