Summer 2020

Cyber physical security of avionic systems

Anusha Thudimilla

Department: Computer Science

CYBER PHYSICAL SECURITY OF AVIONIC SYSTEMS

by

ANUSHA THUDIMILLA

A DISSERTATION

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

2020

Approved by

Bruce M. McMillin, Advisor
Jennifer Leopold
Ricardo Morales
Venkata Siddharth Nadendla
Jonathan Kimball

ABSTRACT

Cyber-physical security is a significant concern for critical infrastructures. The exponential growth of cyber-physical systems (CPSs) and the strong inter-dependency between the cyber and physical components introduces integrity issues such as vulnerability to injecting malicious data and projecting fake sensor measurements. Traditional security models partition the CPS from a security perspective into just two domains: high and low. However, this absolute partition is not adequate to address the challenges in the current CPSs as they are composed of multiple overlapping partitions. Information flow properties are one of the significant classes of cyber-physical security methods that model how inputs of a system affect its outputs across the security partition. Information flow supports traceability that helps in detecting vulnerabilities and anomalous sources, as well as helps in rendering mitigation measures.

To address the challenges associated with securing CPSs, two novel approaches are introduced by representing a CPS in terms of a graph structure. The first approach is an automated graph-based information flow model introduced to identify information flow paths in the avionics system and partition them into security domains. This approach is applied to selected aspects of the avionic systems to identify the vulnerabilities in case of a system failure or an attack and provide possible mitigation measures. The second approach is based on graph neural networks (GNN) to classify the graphs into different security domains.

Using these two approaches, successful partitioning of the CPS into different security domains is possible in addition to identifying their optimal coverage. These approaches enable designers and engineers to ensure the integrity of the CPS. The engineers and operators can use this process during design-time and in real-time to identify failures or attacks on the system.

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my advisor, Dr. Bruce M. McMillin, for the continuous support of my research, and for his immense knowledge, motivation, enthusiasm, and patience. His guidance helped me in all the time of research and writing of this dissertation. I could not have imagined having a better advisor and mentor for my doctoral study.

Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Jonathan Kimball, Dr. Jennifer Leopold, Dr. Ricardo Morales, and Dr. Venkata Sriram Siddhardh Nadendla for their insightful comments and encouragement, and for the inputs they have provided which helped me to widen my research from various perspectives.

I would like to thank my fellow research students: Matthew Wagner, Fernaz, and Simon, for their inputs, feedback, and cooperation. I would also like to thank my entire research group for their thoughtful questions and feedback. Without their passionate participation and input, this research could not have been successfully completed. In addition, I would like to express my gratitude to some of the staff of the Computer Science department: Rhonda Grayson, Elaina Manson, and Dawn Davis, for responding to all my queries.

Also, I would like to thank my friends for accepting nothing less than excellence from me. Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this dissertation. This accomplishment would not have been possible without them. Thank you!

I gratefully acknowledge the funding sources that made my Ph.D. work possible. I was funded by the US National Science Foundation under Award Number CNS-1505610 and received helpful support and funding from the Missouri S&T Intelligent Systems Center.

TABLE OF CONTENTS

Page

ABSTRACT iii

ACKNOWLEDGMENTS iv

LIST OF ILLUSTRATIONS viii

LIST OF TABLES x

NOMENCLATURE xi

SECTION

1. INTRODUCTION 1
   - 1.1. GRAPH-BASED INFORMATION FLOW MODEL - AVIONIC SYSTEMS 4
   - 1.2. GRAPH NEURAL NETWORK MODEL 5
   - 1.3. DISSERTATION STATEMENT 5
   - 1.4. DISSERTATION ORGANIZATION 7

2. LITERATURE REVIEW 8
   - 2.1. CPS SECURITY 8
   - 2.2. INFORMATION FLOW MODELS 9
   - 2.3. CYBER-PHYSICAL SECURITY IN AVIONIC SYSTEMS 10
   - 2.4. OTHER RELATED WORKS 11

3. BACKGROUND 14
   - 3.1. SECURITY MODELS 14
     - 3.1.1. State Machine Model 14

3.1.2. Information Flow Model 15
3.1.3. Non-interference Model 15
3.1.4. Non-deducibility Model 16
3.2. AVIONIC SYSTEMS 16

4. THREAT MODEL 19

5. MULTIPLE SECURITY DOMAIN NONDEDUCIBILITY 21
5.1. MODAL LOGIC 22
5.2. MULTIPLE SECURITY DOMAIN NONDEDUCIBILITY (MSDND) 23
5.3. MULTIPLE SECURITY DOMAIN NONDEDUCIBILITY MODEL (MSDND) 23
5.3.1. Definition: Multiple Security Domain Exclusivity 26
5.3.2. Definition: Event System 26
5.3.3. Definition: Security Domain 26
5.3.4. Definition: Multiple Security Domain NonDeducibility 26

6. GRAPH BASED MSDND 28
6.1. FINDING INDEPENDENT PATHS 29
6.2. ELIMINATE SUBPATHS 29
6.3. VALUATION FUNCTION 31
6.4. SECURITY DOMAINS 32

7. MSDND ANALYSIS 35
7.1. ALTIMETER FAILURE 37
7.2. SATELLITE FAILURE 40
7.3. PITOT STATIC SYSTEM (PSS) FAILURE 44
7.4. PRIMARY FLIGHT DISPLAY FAILURE 48
7.5. MAGNETOMETER FAILURE 50

7.6. RF INTERFERENCE 53
7.7. UNRELIABLE AIRSPEED INDICATIONS 56
7.8. SUMMARY 58

8. MPENN - MESSAGE PASSING EDGE CONVOLUTIONS NEURAL NETWORK 62
8.1. GRAPH NEURAL NETWORKS (GNN) 62
8.1.1. Graph Nets Library 65
8.1.2. Importance of Graphs 68
8.1.3. Message Passing Neural Network (MPNN) 69
8.1.4. Message Passing Edge Convolutional Neural Network (MPENN) 69
8.1.4.1. Message functions 70
8.1.4.2. Graph convolutional layers 70
8.2. EXPERIMENTAL RESULTS 72
8.2.1. Datasets 72
8.2.2. Configuration 73
8.2.3. Metrics 73
8.2.4. Results 76

9. CONCLUSIONS 79
9.1. SUMMARY OF THE PROPOSED APPROACHES 79
9.2. LIMITATIONS 80
9.3. FUTURE WORK 80

REFERENCES 82

VITA 88

LIST OF ILLUSTRATIONS

Figure Page

1.1. CPS Architecture 1

5.1. Aircraft Architecture (1) 24

5.2. Aircraft Architecture (2) 25

6.1. Initial Graph 32

6.2. Information Flow Paths 33

6.3. Security Domains 34

7.1. ATC Helps to Identify the Faulty Altimeter 37

7.2. Graph - Altitude 38

7.3. Graph - Security Domains 39

7.4. Satellite Failure 41

7.5. Graph - Satellite Failure 42

7.6. Graph - Security Domains 43

7.7. Pitot Static System Failure 45

7.8. Graph - Altitude & Airspeed 46

7.9. Graph - Security Domains 47

7.10. Primary Flight Display Failure 48

7.11. Graph - Air Speed, Turn, Heading, Altitude, Vertical Speed, Attitude, Air Speed, Air Density, Outside Air Temperature & Total Air Temperature 49

7.12. Graph - Security Domains 50

7.13. Graph - Attitude & Heading 51

7.14. Graph - Security Domains 52

7.15. ATC Disconnect 53

7.16. Graph - Latitude & Longitude 54

7.17. Graph - Secure and Non-Secure 55

7.18. Graph - Vertical Rate, Airspeed, Vertical Speed and Air Density 56

7.19. Graph - Secure and Non-Secure 58

7.20. Information Flow Paths vs NonDeducibility..... 61

8.1. Message Passing Architecture 68

8.2. MPENN Architecture..... 71

8.3. MUTAG - Accuracy 72

8.4. NCI1 - Accuracy 73

8.5. NCI109 - Accuracy 74

8.6. MUTAG - Loss 75

8.7. NCI1 - Loss..... 76

8.8. NCI109 - Loss..... 77

LIST OF TABLES

Table Page

7.1. Scenario 1 - Information Flow Paths 38

7.2. Scenario 2 - Information Flow Paths 42

7.3. Scenario 3 - Information Flow Paths 46

7.4. Scenario 4 - Information Flow Paths 49

7.5. Scenario 5 - Information Flow Paths 51

7.6. Scenario 6 - Information Flow Paths 53

7.7. Scenario 7 - Information Flow Paths 57

7.8. Information Flow Paths of the Components in Avionic Systems 60

8.1. Results on MUTAG, NCI and NCI109 76

NOMENCLATURE

Abbreviations	Description
CIA	Confidentiality, Integrity, and Availability
CNN	Convolutional Neural Network
CPS	Cyber Physical System
DAG	Directed Acyclic Graph
DFS	Depth First Search
GNN	Graph Neural Network
IT	Information Technology
$MPNN$	Message Passing Neural Network
ND	NonDeducibility
PSS	Pitot-Static System
$SACADA$	Supervisory Control and Data Acquisition
SD	Security Domain
SMC	Statistical Model Checking
SPC	Statistical Process Control
$AHRS$	Attitude and Heading Reference System
ATC	Air Traffic Control
CSP	Communicating Sequential Processes

FMS	Flight Management System
IoT	Internet of Things
RA	Radio Altimeter
RF	Radio Frequency
$HACMS$	High-Assurance Cyber Military Systems
$MSDND$	Multiple Security Domain NonDeducibility

Symbols

$\Box\varphi$	The Modal is always such that ("must be so")
φ	A boolean statement that can be evaluated to *true* or *false* in $w \in W$
V	Valuation Function
$V^i_{S_x}$	Valuation function of entity i w.r.t S_x
$w \vDash \varphi$	Values from world w cause φ to evaluate to *true* ("models")
$w \vdash \varphi$	The statement φ is valid in world w ("yields")
φ	State Variable
S_x	Combination of State Varaibles

1. INTRODUCTION

CPSs are a combination of interacting computing, physical, and human operators controlled and monitored by logic and integrated physics (Pasqualetti *et al.*, 2012). In recent years, there has been a significant increase in the development and deployment of smart and mission-critical computing systems that are characterized by tightly coupled embedded software devices and the physical environment (Humayed *et al.*, 2017). These systems are referred to as critical infrastructures and are ubiquitous in unmanned aerial vehicles, autonomous cars, smart grids, emergency services, oil and gas distribution networks, chemical plants, manufacturing, communications, health care, and data centers. The security and availability of the critical infrastructures are of utmost importance, and failure in ensuring security and availability could have a catastrophic impact on economic growth and public safety.

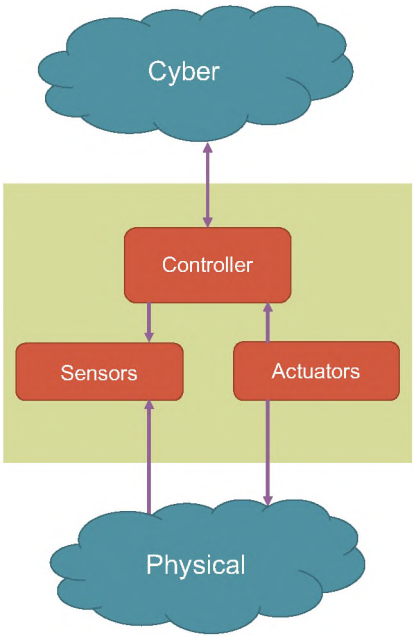


Figure 1.1. CPS Architecture

One of the key features of the CPS is the strong integration of cyber and physical processes. The physical environment plays a crucial role in these systems by providing necessary information to achieve significant functionalities. The integration of the physical and cyber worlds is achieved through sensors that constitute the physical world; controllers that render sensory inputs for cyber and physical elements, and actuators that drive the physical world via controller inputs in the physical world (see Figure 1.1) (Sampigethaya and Poovendran, 2013). Due to tight coupling between the cyber and the physical components, new forms of risk have been introduced in the CPS. These risks have not been considered adequately in the existing computing domain due to the lack of availability of tools to identify vulnerabilities that arise due to the complex interactions between the cyber and physical worlds. These risks can be classified as (1) cyber elements affecting the physical environment and (2) physical elements affecting the cyber components. These interactions lead to new security issues as assessing vulnerabilities and threats becomes challenging. In addition to these complex interactions, the complexity of the CPS itself presents more challenges in devising security mechanisms. Therefore, a concrete understanding of cyber-physical interactions, threats, vulnerabilities, and attacks is essential to the development of efficient security mechanisms. Hence, assuring the confidentiality and integrity of information in a cyber-physical system is of prime importance.

Various security mechanisms have been deployed in the industry to identify vulnerabilities and threats in CPSs. As each CPS infrastructure is different from others, different security models are designed to provide suitable solutions. One of the widely used methods to address the security vulnerabilities and threats are formal methods. Formal methods serve as a framework for the specification, design, and verification of software-intensive embedded systems, with a focus around provable functional safety and security targeted at CPS (Clarke and Wing, 1996). They are a widely used means to provide the capability to specify, model, verify, and integrate modern CPSs, and to design flows for enhancing functional safety and security. Formal methods are considered to be one of the most re-

liable mechanisms to help achieve security and privacy in CPSs. They model embedded systems and adversaries in order to prove that a system under consideration is immune to various classes of attacks. They provide well-established scientific foundations in the form of precise adversary and system models and derive sound conclusions about the possible behaviors of the system as the adversary interacts with it. Formal methods work well when an adversary's behavior and the implications of an attack are well known.

Another popular security model to address vulnerabilities and threats in a CPS is the Byzantine model. This model considers a set of connected abstract machines as a graph in which few nodes may be faulty, and the communication link between these nodes to transfer messages are represented by formal expressions. Recently, there have been substantial progressions in automated and semi-automated formally verified analysis methods, such as the *seL4* microkernel (Elkaduwe *et al.*, 2008), the *CompCert* optimizing C compiler (Leroy *et al.*, 2012), and secure vehicles and drones emerging from DARPA's High-Assurance Cyber Military Systems (HACMS) program (Fisher, 2014).

With the growth in CPSs, the above mentioned approaches will not be feasible enough for two reasons:

- Any change or a new addition to the CPS will result in updating the security policy, and this could become tedious over time.
- As each CPS is attributed with different characteristics, providing a universal and scalable solution that can be applied to any CPS is difficult.

In this dissertation, two novel approaches have been introduced: (i) graph-based information flow analysis model and (ii) a graph neural network model.

1.1. GRAPH-BASED INFORMATION FLOW MODEL - AVIONIC SYSTEMS

The graph-based information flow model is aimed at identifying the vulnerabilities in a CPS and providing mitigation measures. A study has been performed on one important CPS application of avionic systems to ensure the safety of the onboard cyber and physical flight systems as well as ground communication.

Aviation systems are a combination of many physical world components, such as electronics, hardware, infrastructure, and humans, which are heavily reliant on digital computing, storage, software, or data networking to function efficiently. Such components are becoming the heart of modern aviation infrastructure and are consuming and sourcing information at all times. Modern aircraft are advancing by incorporating the "cyber" layer in physical components, infrastructures, and humans for improving onboard system performance and services. The cyber layer must not only enhance performance but also enable it. An in-depth analysis of all cyber-physical properties, integrations, and risks is essential to ensure the security of the CPSs.

Advancements in technology bring new challenges for those involved with the process and maintenance of modern passenger aircraft. The consequences of software failure can range from insignificant (no effect on aircraft performance) to catastrophic (e.g., major avionic system failure and engine faults). With the advancements in communications and the quality of jet engines, there is an increased shift towards automation. However, automation creates additional problems such as pilots finding it increasingly difficult to cope with a lack of transparency of onboard systems which can lead to a loss of situational awareness and loss of control in flight (LOC-I). Loss of control has been the primary cause of accidents during the years 2010 and 2014 in which 43% of the 37 fatal accidents was due to pilot's lack of understanding of what was happening in the cockpit.

The limitations experienced by modern air transport system include loosely integrated flight phases in time and space; voice communications between pilot and controller; predeparture operations based on manuals; and trajectories based on clearances not adapt-

able to varying factors like failures, emergencies, congestion, and weather. These processes can be optimized by cyber-physical interactions using a range of solutions: trajectory-based flight operations, satellite-based navigation, communicating monitoring and position data, automated data links between pilot and controller, and a strongly connected information network of real-time air traffic control and meteorological data (Sampigethaya and Poovendran, 2013). The common impacts of cyber and physical interactions that affect performance must be balanced, as there are trade-offs to be measured. Additionaly, the Internet of Things (IoT) has introduced new challenges by making aircraft devices and sensors a standard rather than nice-to-have options. This makes the maintenance of secure data links an important factor for the pilots as well as the in-flight passengers. Aviation security has moved from the position of seldom-discussed to that of critical importance. With the evolving changes and the introduction of new systems, aviation security remains a critical area of research to improve security measures.

1.2. GRAPH NEURAL NETWORK MODEL

A next-generation CPS requires the implementation of advanced deep learning algorithms to incorporate intelligence into the physical processes to enable high-performance computing architectures to identify meaningful patterns. In this dissertation, a novel deep learning method has been introduced to classify the graphs of a CPS based on the attacks or failures in a CPS.

1.3. DISSERTATION STATEMENT

Information flow analysis supports the resilient design and active detection of anomalies in CPSs. The complex nature of CPSs significantly increases the difficulty of determining information flow as well as mitigating the corresponding integrity problems. This work applies graph based analysis to CPSs to determine information flow and partition them into

security domains based on the reliance of information flow paths. In this dissertation, two approaches have been proposed: (i) the first approach is based on graph-based analysis, and (ii) the second approach is based on graph neural networks.

The results show that the inter-dependency between the physical and cyber properties of a CPS can preserve and leak information. The key to formalize these approaches is to determine a consistent semantic representation of the cyber and physical elements, their interaction, and the physics of the system, and to devise a formal modeling technique for determining information flow. This dissertation presents a formal model for information flow analysis in a CPS and introduces an approach to perform the analysis, including both state-based analysis and automated analysis through graph networks. An avionics system, which is a critical infrastructure, is used to demonstrate the first approach, whereas various benchmark datasets from the chemical domain are used to demonstrate the second approach. The proposed approaches can analyze the information flows, verify whether the avionic system and the datasets used inherently preserve integrity, and categorize the security space into different security partitions based on the integrity level. The ultimate goal of these partitions is to identify the optimal number of security domains that illustrates the coverage of the security domains.

This dissertation attempts to represent the CPS as a graph-based network and find the optimal number of security partitions that define the trust level of the system. This is achieved by using two approaches:

- Graph-based analysis: This approach is used to identify the information flow paths and cluster them into security partitions.

- A neural network-based graph classification: This approach uses message passing neural network and edge convolutions to classify the graphs as either secure or non-secure based on the edge and node attributes.

1.4. DISSERTATION ORGANIZATION

Section 2 describes the related work performed in the field of cyber-physical security, introducing the reader to the existing work done in this field and the associated shortcomings of implementing the existing methods. To illustrate the implementation of the automated analysis, Section 3 presents a brief overview of the avionic systems and information flow models. Section 4 presents the threat model used to identify attacks, vulnerabilities, threats, and countermeasures that impact the system. The graph-based information flow model is introduced in Section 5 and is applied to selected aspects of the avionic systems to identify the security vulnerabilities and propose mitigation techniques if applicable. Section 8 presents the graph neural network model implementation and experimental results. Section 9 discusses the summary of this work, limitations, and future work.

2. LITERATURE REVIEW

In this section, a review of some of the previous work is introduced. This review helps in demonstrating the idea of information flow and also presents a few shortcomings that this work aims to avoid.

CPSs play an essential role in controlling and monitoring various control processes in critical infrastructure systems. Due to the complex nature of CPS interactions, there is a necessity to devise mechanisms to protect against attacks and failures. There is an increase in the analysis of vulnerabilities of CPS resulting from malicious, faulty components, and system failures. Various security methods have been proposed to address security issues such as the denial of service attacks (Amin *et al.*, 2009), false data injection attacks (Liu *et al.*, 2011), stealthy deception attacks (Teixeira *et al.*, 2010), and replay attacks (Mo and Sinopoli, 2009) in critical infrastructures such as avionics, smart-grids, and autonomous vehicles.

2.1. CPS SECURITY

Research investigations have been carried out over the past decade that specify the importance of CPS security. Earlier works such as (Cárdenas *et al.*, 2011, 2008a,b; Sandberg *et al.*, 2010) reiterated the fact that the concept of cyber attacks is not only significant to the cybersecurity community but also to many other interest groups. Instead, these threats must be analyzed from an extensive system and infrastructure perspective, and that component-wise approaches may not be sufficient and need to be studied and analyzed from infrastructure-wide and system-wide perspectives (Dibaji *et al.*, 2019). Deception attacks, disclosure attacks, and disruption attacks have been discussed at length in (Cárdenas *et al.*, 2008b; Teixeira *et al.*, 2015). CPS security issues arise in a wide-range of applications. On a regular basis, there are reports of cyber attacks in various sectors that include a

cyber component. A large number of subsystems present in power systems indicates that the impact of cyber-physical attacks varies significantly depending on when and where they occur. In addition to these applications, CPS security was studied in (Cho and Woo, 2017) to address the cyber attack in 2014 in nuclear power plants and analyzed various security methods to identify such attacks. In (Wang *et al.*, 2016), a review on deception and disruption attacks in CPSs was performed. The importance of security in Supervisory Control and Data Acquisition (SCADA) was discussed in (Miller and Rowe, 2012) and in Modbus control systems in (Huitsing *et al.*, 2008). A survey was carried out by (McLaughlin *et al.*, 2016) on security aspects of information and communication channels of industrial control systems. The survey done by (Lamnabhi-Lagarrigue *et al.*, 2017) on industrial control systems also signified the importance of security. Many researchers have used mathematical models that ed the physical systems and were composed of a set of relations between the system states and its control inputs. Control inputs are restricted by physical inputs to define a finite set of admissible inputs or controls. A deviation in these inputs results in inaccurate outputs, which in turn affects the system.

2.2. INFORMATION FLOW MODELS

Information flow security is yet another widely used approach to prevent secret data from leaking to malicious entities. Two primary variants of information flow security are either static or dynamic. The dynamic approach uses labels to describe the security level and propagates these labels to ensure the integrity of the data with respect to invariants or predefined policies, whereas the static approach executes information security policies. A vast majority of research in this field is focused on proving the non-interference property (describes the information flow restrictions) and using a combination of language features and system models to implement information flow security (Sabelfeld and Myers, 2003).

In the literature, various information flow models based on process algebra have been introduced. In (Lowe, 2004) the author introduced a new definition of information flow based upon an operational model of communicating sequential processes (CSP), which analyzes how the behavior of one entity or agent can influence another entity's or agent's view of the system. There is an extensive amount of research done in the area of CPS security to address the security issues by using information flow paths. The real challenge for information flow security is applying the vast theory and language-based designs such as Jif (Myers *et al.*, 2001) and FlowCaml (Pottier and Simonet, 2002; Simonet and Rocquencourt, 2003) to real-world problems (Zdancewic, 2004). Widely known methods to perform security analysis are formal methods that consider the cyber-physical interactions to identify potential risks and challenges that arise due to the complex transformations in airplane systems.

2.3. CYBER-PHYSICAL SECURITY IN AVIONIC SYSTEMS

The research in ADS-B security either focuses on cyberattacks or physical component failures, but not both. Significant research has been done in this field to analyze the security vulnerabilities, attacks, and system failures in avionic systems (Manesh and Kaabouch, 2017; McCallie *et al.*, 2011; Strohmeier *et al.*, 2014). A time-stamp based method based on signal propagation time to identify and reject spoofed ADS-B messages between senders and receivers was proposed by (Kim *et al.*, 2017). A light weight security solution to guarantee the privacy and integrity of ADS-B messages by integrating crypto-primitives such as FFX and TESLA was proposed by (Yang *et al.*, 2017). Similar work was carried out in (Yang *et al.*, 2018) for congested data links and resource-constrained avionics. In (Thudimilla and McMillin, 2017), ProVerif was used to identify the attacks in ADS-B and TCAS systems. The analysis was limited to proving observational equivalence (anonymity property) through the composition of processes, thus lacking computational

capability. Several works have been done to analyze the security vulnerabilities, attacks, and system failures in ADS-B systems (Manesh and Kaabouch, 2017; McCallie *et al.*, 2011; Strohmeier *et al.*, 2014) that consider cyber attacks alone.

2.4. OTHER RELATED WORKS

Anomaly detection in CPSs has been extensively studied. Practical problems with this approach include the need for precise knowledge of the system's design and configurations as well as the need to accurately model the system's complex physical behavior. The authors in (Mitchell and Chen, 2014), categorized anomaly detection methods as knowledge and behavior-based methods. Knowledge-based detection techniques examine well-known attack characteristics but require keeping a dictionary of attack signatures. Despite having low false positive rates, these methods are ineffective against zero-day attacks. In contrast, behavior-based techniques check for anomalies in runtime behavior. These techniques are pervasive in CPS anomaly detection, since CPSs are automated and present more uniformity and predictability than conventional information technology (IT) systems (Kravchik and Shabtai, 2018).

There are multiple anomaly detection techniques such as statistical process control (SPC) (mur, 2016) that include shewhart charts, exponentially weighted moving average, and cumulative sum. These methods are inadequate to handle the complex nature of the CPS as CPS operates in an unpredictable environment. Consequently, this has resulted in an increased shift from the formal specification or signature-based techniques to the machine learning techniques to develop rational and adaptive approaches to identify anomalies in CPSs (Kravchik and Shabtai, 2018). Nevertheless, the implementation of machine learning techniques in detecting anomalies in data is considered a challenging task. This is mainly due to the dependency of these methods on enormous amounts of labeled data and the anomaly classes to learn from, in which inconsistencies are usually rare or neglected in

the real environment. In addition, most of the existing unsupervised methods may not be effective to recognize anomalies due to the existence of temporal dependencies and noise in real-time applications.

Below are some of the tools used for CPS security in the literature.

- *UPPAAL:* UPPAAL is an integrated tool environment for modeling, validating, and verifying real-time systems modeled as networks of timed automata (Behrmann *et al.*, 2004). Few functionalities of the UPPAAL extension include visualization of the results in the form of probability distributions, time-bound based evaluation of the number of runs, and computation of expected values.

- *Statistical Model Checking (SMC):* SMC refers to a set of techniques that monitor several runs of a system concerning the specific property. SMC utilizes results from statistics to get an overall estimate of a design's correctness by approximating undecidable problems. This approach has been widely used in model checkers. Traditional SMC model checkers described in (Katoen *et al.*, 2011; Sen *et al.*, 2005; Younes, 2005) are inadequate to handle timed systems and are limited in either addressing the capability of computing an estimate of the probability or testing whether this probability is greater or equal to some threshold.

- *Petri nets:* Petri nets are a mathematical and graphical modeling tool that describes the structure and behavior of a system by modeling distributed causality and concurrency of a system (Peterson, 1977). Petri nets have been widely used in manufacturing systems, fault-tolerant systems, formal languages, and communication networks.

- *MoDeST:* Another framework that is used for model checking is the MoDeST toolset (Bohnenkamp *et al.*, 2004) which handles the specification of stochastic timed systems but fails to yield full stochastic models in case of parallel composition.

There is a necessity to address the security vulnerabilities that arise from the complex interactions between the cyber and physical components. This dissertation is aimed at addressing the above-specified issues by focusing on cyber and physically enabled attacks by using an automated model-based approach to identify security risks and providing mitigation measures.

3. BACKGROUND

This section presents a brief overview of the various security models and avionic systems used in this dissertation.

3.1. SECURITY MODELS

Security models are used to determine the implementation of security, agents that can access the system, and the objects that they can access. To be precise, they are a way to validate security policies. In general, security models are implemented by enforcing confidentiality, integrity, or availability (also referred to as CIA triad).

- *Confidentiality:* Confidentiality is the term used to ensure privacy. Confidentiality plays a vital role in preventing sensitive information from being leaked to unauthorized personnel.

- *Integrity:* Integrity is one of the essential elements of security objectives. Integrity plays a vital role in security because it can verify the modification of data by unauthorized users, unauthorized changes made by authorized users, and ensures consistency of data internally as well as externally.

- *Availability:* The system's availability to authorized users is vital for an information system to be useful. Availability ensures uninterrupted access to the system. Threats to availability include non-malicious threats such as system failures and malicious threats such as denying access to users.

3.1.1. State Machine Model. The state machine model is based on a finite state machine and is used to model complex systems (Conrad *et al.*, 2012). They deal with different attributes, such as actions, transaction functions, and state variables. The state

machine defines the behavior of a finite number of states, the transitions from one state to another state, and actions that result from the transitions. A state machine model prevents the system from entering into a vulnerable state by monitoring the system's state. The state machine model is considered a foundation for many security models as it relays the state of the system at a given time.

3.1.2. Information Flow Model. The information flow model is based on the state machine concept and serves as the basis for designing popular security models such as the Biba, Bell-LaPadula, non-interference, and non-deducibility models. The information flow model is composed of objects, state transitions, and states. The ultimate goal of this model is to prevent insecure information flow and unauthorized access.

3.1.3. Non-interference Model. The non-interference model defined in (Goguen and Meseguer, 1982) states that "objects and subjects of different levels do not interfere with the objects and subjects of other levels. The model uses inputs and outputs of either low or high sensitivity. Each data access attempt is independent of all others, and data cannot cross security boundaries." In terms of CPS modeling, objects refer to subsystems, components, or documents, whereas subjects refer to operators, system users, networks, applications, or processes. In this model, a system's sensitivity levels are categorized as low or high. Low sensitivity represents unclassified information, whereas high sensitivity represents classified information or resources that require specific clearance to access. It essentially states that users with low sensitivity cannot learn information about the user's activities with high sensitivity.

With the non-interference model, a strict separation of various security levels can be achieved by hiding activities with high sensitivity irrespective of what lower-level users can see or access. This separation ensures minimizing leaks and breaches that might occur through secret channels. This model considers the impact of higher security level subjects' activities on the system's state, keeping lower security levels separate. Hence, creating covert channels through shared resources or inference attacks is not possible.

3.1.4. Non-deducibility Model. In 1986, David Sutherland (Sutherland, 1986) introduced the information-theory based non-deducibility security model to quantify the information flow from one user to another. Sutherland's definition states that "Given a set of possible worlds Ω and two information functions f_1 and f_2 with domain Ω, we say that information flows from f_1 to f_2 if and only if there exists some possible world ω and some element z in the range of f_2 such that z is achieved by f_2 in some possible world, but in every possible world ω' such that $f_1(\omega') = f_1(\omega), f_2(\omega') \neq z$". Each discrete execution of the system can be considered an element of the set of possible worlds. Information about the system is represented by an information function whose domain is the set of possible worlds. This model is considered to be too restrictive and weak as the information can flow from high-level objects to low-level objects but the reverse is not possible.

3.2. AVIONIC SYSTEMS

MSDND is applied to selected aspects of the avionic systems to check for failures/attacks and then partition them into different security domains based on the behavior they exhibit.

The aviation industry experienced exponential growth in computing capabilities and interaction complexity. Current practice addressing safety concerns include a build and test approach following industry-standard recommended practices such as SAE ARP 4761 and 4754. The software development practices furnish limited attention to nonfunctional qualities such as timing, latency, performance, reliability, safety, or security. These concerns are usually addressed by modeling and analysis or simulation, which are captured by analytical models that quickly become outdated as the architecture and design emerge. This results in delayed discovery of system-level errors, with studies showing up to 80% leakage to this phase.

Modern aircraft are heavily reliant on the extensive use of electronic instruments and displays. The significant advantage of using electronic devices is the ease of exchange of data between different devices. It serves as a basis for not relying on the flight manuals and provides improved automatic flight control.

Modern aircraft use increasingly sophisticated avionic systems that are composed of heterogeneous CPSs that assist in controlling and operating the plane by the crew. These systems use microprocessor-based computer systems and are composed of hardware and software components that are capable of processing large amounts of data in a short time. Avionic systems are deployed in a wide variety of applications, such as flight control and instrumentation systems, navigation, and communication systems. These systems function by receiving data from multiple sensors, including air temperature probes, fuel sensors, angle-of-attack probes, and pitot-static pressure systems. These computers process data from the sensors, apply various functions and forward information to the electronic displays. The pilots continuously monitor the status of the engine and environment from these displays. These systems collect information from many interacting components to ensure the proper functioning of the system.

Most integrated multi-sensor applications receive information from multiple sensors to provide redundancy management, improve performance, achieve graceful degradation when sensor failures occur, and increase robustness. Flight operations that provide guidance information to the pilot depend primarily upon precise and continuous awareness of flight position and attitude. Technical requirements for air navigation systems primarily include accuracy, physical attributes such as weight and volume, electrical power, and system integrity.

A detailed description of the various components used in scenarios described in Section 7 can be found in Administration (2016, 2017). Flight instruments enable an aircraft to be operated with maximum performance and enhanced safety, especially when flying long distances. Manufacturers provide the necessary flight instruments, but to use them

effectively, pilots need to understand how they operate. Pilots need to be very familiar with the operational aspects of the pitot-static system and associated instruments, the vacuum system and associated instruments, the gyroscopic instruments, and the magnetic compass.

4. THREAT MODEL

Before coming up with a security solution, having the knowledge of who/what we protect a CPS from is of great importance in addition to the knowledge of existing attack mechanisms, vulnerabilities, and failures. The potentiality aspect is critical in this context; potential threats that may not necessarily have occurred, but might. The loss might be in safety measures, confidentiality, integrity, safety, or availability of resources, whereas the harm implies harming people, the environment, or systems. As CPS applications are becoming more ubiquitous, people are becoming a critical asset to protect, in addition to the information and communication elements.

Building a good threat model is crucial to assess the security vulnerabilities in the system and coming up with efficient mitigation measures. To launch an attack an adversary must compromise one or more components in the system or introduce malicious components into the system. The threat model is composed of the following:

- *Source:* The source of a threat is the entity responsible for initiating a threat which includes system failure, adversarial attacks, and environmental factors.
  - Failures: This class of threats represents an observable error in the system that is caused accidentally or by malfunction of a legitimate CPS component.
  - Adversarial: This class of threats represents a malicious outside attacker or an insider with an intention to compromise the system. Few examples include terrorists, activists, and criminal groups trying to compromise the system by injecting malicious data or tampering the existing data. Adversaries can be passive or active. Passive adversaries have limited capabilities such as eavesdropping on communication channels, whereas active adversaries have the ability to modify the contents of the communication channels and compromise the data.

  - Environmental: This class of threats represents failures or malfunctions caused by natural disasters.

- *Goal*: The ultimate goal of the threat model is to capture the features of the system that may lead to system failure or identify the features that are modified as a result of an attack.

- *Consequence*: The attack or failure in a CPS results in compromising integrity, safety, and availability of the system. In this threat model, attacks and failures are considered to exhibit similar behavior (Stroud, 2003).

In case of an attack, the adversary is assumed to have full control of the faulty components: the adversary can eavesdrop, intercept, and modify any message with respect to the faulty component. In this section, failures and attacks are assumed to be arbitrary and unbounded. The threat model assumes that the adversary cannot exploit certain aspects of the systems such as:

- The adversary cannot corrupt or modify the proposed model.

- The adversary cannot modify more than half of the participating entities to perform specific operations.

5. MULTIPLE SECURITY DOMAIN NONDEDUCIBILITY

Traditional security models such as the Bell-LaPadula model, non-interference, and non-deducibility are better expressed in terms of Shannon-style information flow (McLean, 1990). Extending this work, researchers have introduced flow-based security models by analyzing security-relevant causal factors. Models such as non-interference, Generalized non-interference, and extensions to non-interference designed to protect high-level output, are not flexible in considering causal factors as they require to consider programs as explicit input to systems. The Bell-LaPadula model that primarily addresses access control has few drawbacks such as the model addresses only confidentiality and does not address access control over covert channels (Gallagher, 1992).

Early research in security models focused on the nondisclosure of information. There has been a significant shift towards integrity models due to the importance of data in decision-making and actions (McCumber, 1991). A CPS's safety can be compromised by a failure to understand requirements or defects in its implementation. In terms of security, defining what a system should do is comparatively a difficult task as a high degree of precision is required to prevent undesired outcomes or defects. To increase the model's precision, researchers have used formal language or information-theoretic approaches to model CPSs.

This shift toward information-theoretic models of security introduces new opportunities to develop security models that have significant dependence on information flows. One primary class of security models that are proven to be the closest match to this approach is the non-deducibility property. This property is essentially based on a theory of information sharing, which attempts to ensure security by partitioning the security space into just two partitions or domains: high and low. On the other hand, non-interference based models include information flows inherently, but there has been little work done in developing a

method for evaluating such models. Will these absolute partitions guarantee to identify the inconsistencies in the system? Will these partitions help in identifying attacks in a complex CPS with more than two security partitions? To address these issues, this section presents an improved version of non-deducibility called Multiple Security Domain NonDeducibility (MSDND), which considers multiple partitions.

5.1. MODAL LOGIC

The applications of modal logic to mathematics and computer science have become increasingly essential (Goldblatt, 2006). Modal logic refers to the validation of an expression in the form of 'necessarily' and 'possibly'. The term 'modal logic' is used more broadly in the security domain, which is a family of logics to characterize the difference between valid and invalid arguments. Creating a logic is a difficult task as it is dependent on a set of axioms and rules designed to prove the valid arguments statable in the language. The main challenge lies in achieving soundness and completeness by formulating every valid argument as proof in the system. Modal logic is an umbrella term for a variety of classes, such as provability logic and advanced modal logic.

Modal logic is primarily used as a tool for representing structures or models. Modal logic talks about algebraic semantics, topological semantics, and graphs. These are known as Kripke semantics for modal logic and is the best-known style of modal semantics. In the literature, modal logic has been used as a tool for reasoning about time, beliefs, computational systems, necessity, and possibility (Blackburn and Van Benthem, 2007). Although diverse, these applications have some critical things in common such as time-flow, relations between cognitive alternatives, computational state transitions, and networks of possible worlds. All these can be represented as simple graph-like structures, and modal logic is proved to be an appealing tool for handling such structures as they are rich in representing information.

5.2. MULTIPLE SECURITY DOMAIN NONDEDUCIBILITY (MSDND)

MSDND was introduced in (Howser and McMillin, 2013), which is based on modal logic (Blackburn *et al.*, 2006; Goldblatt, 2006) to address the shortcomings of the traditional security models which work well only if the boundaries for the security domains are clearly defined. The MSDND model presented in (Howser and McMillin, 2013) can check for non-deducibility with respect to two states at any single point of time. Calculating the security domains for complex infrastructures are difficult. However, our work is aimed at handling this issue as well as automating the MSDND analysis for partitioning the security domains based on the information flow traversal.

In this dissertation, *{V}* is used to define a set of valuation functions, such that $V^{i}_{S_x}(w)$ indicates the value of state variable S_x as seen by an entity i in world w. **NOTE:** If the state variable, say S_i has no valuation function, which returns the value of a state variable, then MSDND fails to determine the value of that state variable nor the value of any logical expression associated with the state variable (Howser and McMillin, 2013).

Note: State variables are represented by $\varphi_0, \varphi_1, ...\varphi_m$ or by a combination of state variables represented by S_x.

$$S_x = \varphi_0 \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge ....$$

In this paper, each state variable is associated with the component ID (See Figures 5.1 and 5.2). For example, Satellite 1 is represented by φ_0, Barometric Altimeter by φ_6 and control panel by φ_9.

5.3. MULTIPLE SECURITY DOMAIN NONDEDUCIBILITY MODEL (MSDND)

This section discusses the MSDND model to check for non-deducibility with respect to multiple states and overlapping security domains. It also introduces various definitions that are used to define MSDND.

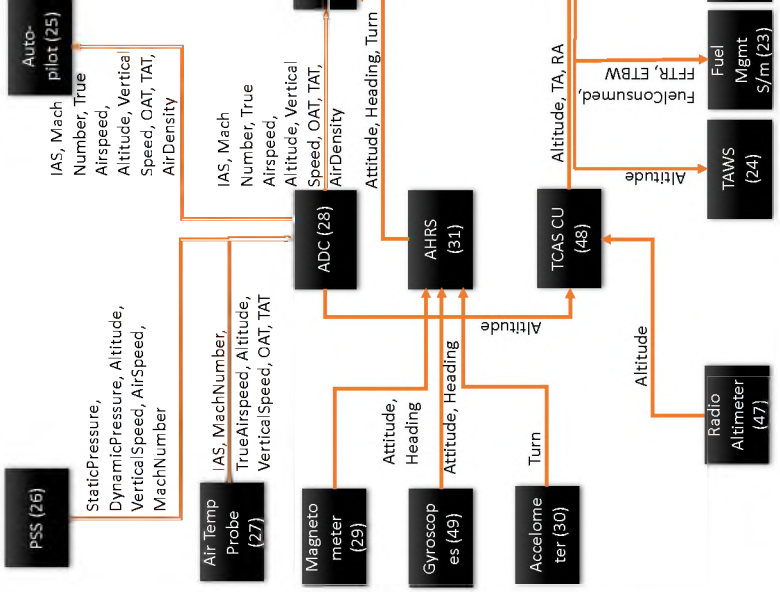
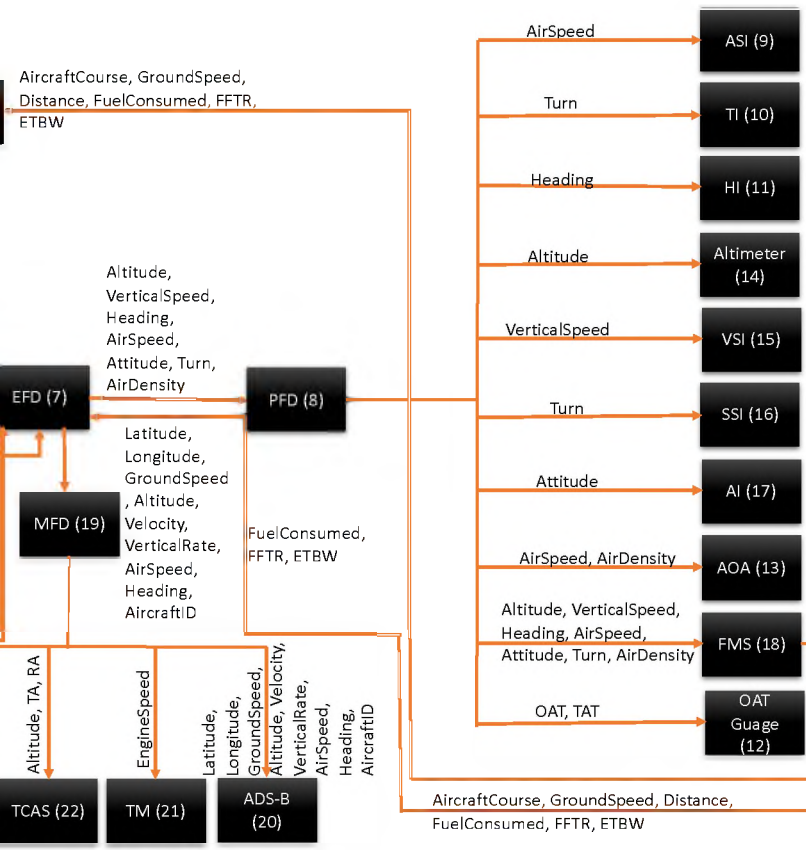


Figure 5.1. Aircraft Architecture (1)


AirSpeed
ASI (9)
AircraftCourse, GroundSpeed, Distance, FuelConsumed, FFTR, ETBW
Turn
TI (10)
Heading
HI (11)
Altitude
Altimeter (14)
Altitude, VerticalSpeed, Heading, AirSpeed, Attitude, Turn, AirDensity
VerticalSpeed
VSI (15)
EFD (7)
PFD (8)
Turn
SSI (16)
Latitude, Longitude, GroundSpeed, Altitude, Velocity, VerticalRate, AirSpeed, Heading, AircraftID
Attitude
AI (17)
MFD (19)
FuelConsumed, FFTR, ETBW
AirSpeed, AirDensity
AOA (13)
Altitude, VerticalSpeed, Heading, AirSpeed, Attitude, Turn, AirDensity
FMS (18)
OAT, TAT
OAT Guage (12)
Altitude, TA, RA
EngineSpeed
Latitude, Longitude, GroundSpeed, Altitude, Velocity, VerticalRate, AirSpeed, Heading, AircraftID
TCAS (22)
TM (21)
ADS-B (20)
AircraftCourse, GroundSpeed, Distance, FuelConsumed, FFTR, ETBW

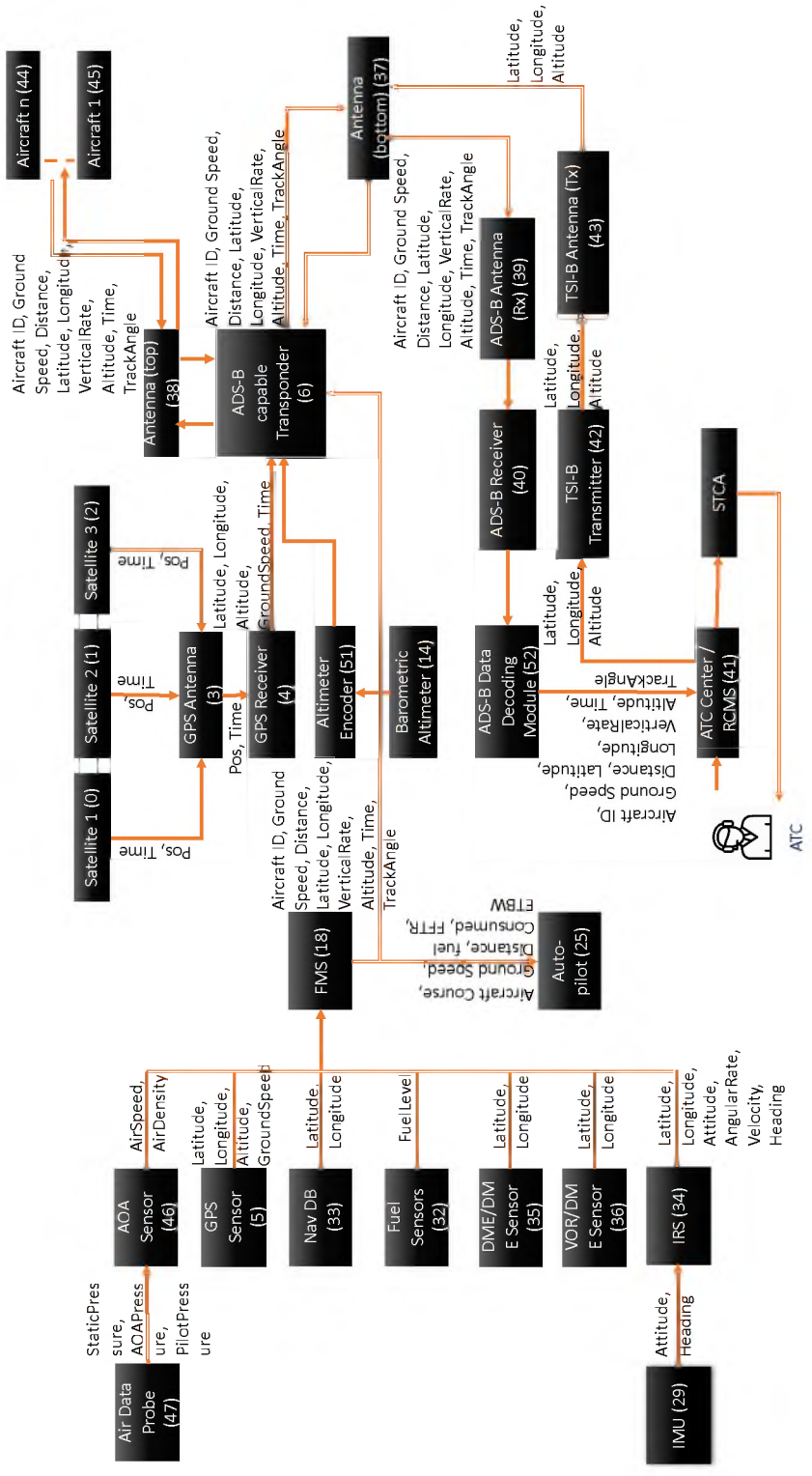


Figure 5.2. Aircraft Architecture (2)

5.3.1. Definition: Multiple Security Domain Exclusivity. There exists some world with multiple states in which, at any instance, the system can be in one true state, and the others are false.

$$f(S_a, S_b, S_c, ...) = \begin{cases} where\ one\ of\ S_a, S_b, S_c, ...\ is\ True \\ otherwise\ False \end{cases} \tag{5.1}$$

5.3.2. Definition: Event System. Event System (*ES*) is a composition of a different sets of objects and the events that are passed from one object to another based on the inputs. The shift of these objects and events triggers the transition of one state to another state.

5.3.3. Definition: Security Domain. A security domain is a logical partition of the *ES* based on different parameters depending on the context of entity *i*. Each security domain is composed of different states.

$$\cup U_{i \in I} SD^i = (ES) \tag{5.2}$$

5.3.4. Definition: Multiple Security Domain NonDeducibility. In the MSDND model, an entity *i* is any part of the system capable of independent observation or action. The *ES* consists of multiple security domains, SD^i, as viewed by each entity *i* in the model. These domains may, or may not, overlap depending on the complexity of the *ES* (Howser and McMillin, 2013). A system is MSDND if

$$\begin{aligned} MSDND(ES) = \exists w \in W: [w \vdash \Box f(S_a, S_b, S_c, ...)] \\ \wedge [w \vDash (\nexists V^i_{S_a} \wedge \nexists V^i_{S_b} \wedge \nexists V^i_{S_c} ...] \end{aligned} \tag{5.3}$$

An MSDND proof creates a logical argument of conditions on the observable state of the system under consideration. These conditions are assessed for their valuation from the point of view of a particular security domain. The valuation function determines the validity of a particular state based on the validity of the preceding states. If no valuation

function can be found, then the system is MSDND secure (which is a bad thing as it means an attacker can hide their actions from a particular security domain). Breaking MSDND is a good thing as it means the system can detect the attacker.

As mentioned earlier, we use a variant of MSDND modal (Howser and McMillin, 2017) to model complex CPSs and analyze the security vulnerabilities. MSDND is used to model avionic systems to identify the vulnerabilities and propose mitigation techniques.

6. GRAPH BASED MSDND

The analysis used in this work considers an aircraft system as a graph network where each node represents a component in the system, and the edges represent the information flow between two nodes. Each edge consists of a set of labels, and each label has a value associated with it. This algorithm consists of five steps:

- Identify all the paths in the network using DFS with respect to a label, which indicates the information flow from one node to another and sorts the result set in descending order of subgraph size.
- Identify the subgraphs and eliminate them to get a reduced unique subgraph set.
- In the reduced set, traverse through each edge to check for discrepancies indicated by the inconsistent values associated with the edge labels during run-time and design time. The discrepancies in the data are caused by an attack or a failure in the system. These discrepancies can be identified by finding the in-degree for each node and check for the consistency of the values associated with labels.
- A node with in-degree ≥ 3 will help break the NonDeducibility, as it contains more than two information flow paths, which helps identify the faulty component or the component under attack, which is responsible for sending incorrect data.
- Classify the information flow paths into *Secure* and *Non-Secure* domains based on the valuation function.

A graph $G = (V, E, L)$ consists of a set of nodes V, a set of edges E, and a set of labels L associated with each edge. A graph $S = (V_s, E_s, L_s)$ is a subgraph of graph $G = (V, E, L)$ iff $V_s \subseteq$ V, $E_s \subseteq$ E and L_s contains l where l is the label under consideration.

Figure 5.1 and Figure 5.2 represents various interacting components in an aircraft referred as nodes. Each edge has a set of labels and values associated with them, which represents the information flow.

Algorithms 1 and 2 are used to find all the graphs associated with each label and eliminate the subpaths in order to obtain unique graphs for each label. Source code for these algorithms can be found here: https://github.com/anushaat/MSDND

6.1. FINDING INDEPENDENT PATHS

Algorithm 1 uses DFS to find all the paths with edges labeled with l. Theorem 6.1.1 presents the logical arguments that are necessary to prove the correctness of Algorithm 1.

Theorem 1: In DFS of a directed acyclic graph (DAG) $G = (V, E, L)$, vertex s is a descendant of vertex d iff the search discovers d, that there is a path from s to d consisting entirely of edges with label l.

Proof: Suppose that DFS is run on a DAG $G = (V, E, L)$ to determine the independent paths for each vertex $v_i \in$ V. It suffices to show that for any pair of distinct vertices $s, d \in$ V, if G contains an edge from s to d. If $s = d$, then the path from s to d contains only s which indicates the source or initial node. If d is an immediate descendant of u, then the path from s to d contains label l. If d is any descendant of s, all edges on the simple path from s to d contains label l.

With a runtime complexity of $\mathcal{O}(V + E)$, where V represents the number of nodes and E represents the number of edges, this algorithm will result in all the subpaths associated with the specified label.

6.2. ELIMINATE SUBPATHS

This algorithm is aimed at eliminating the frequent subpaths which occur more than once. This helps in achieving non-redundant subpaths, which helps in identifying the faulty component or the component under attack. Theorem 6.2.1 provides necessary logical arguments to prove the correctness of Algorithm 2.

Algorithm 1 Algorithm to find independent paths

Input: Graph dataset G, find all the paths with respect to label l
Output: Result set R which includes all paths associated with each label l

```
1: function DFS(G, u)
2:     visited.add(u)
3:     if onPath.contains(u) then
4:         result.add(Path)
5:     else
6:         for all v ∈ adj[u] do
7:             if v.labels.contains(l) then
8:                 DFS(G, u)
9:     Path.pop()
10:    return result
```

Algorithm 2 Algorithm to eliminate the subpaths

Input: Graph dataset G, eliminate frequent subpaths
Output: Set of all graphs associated with each label l

```
1: function EliminateSubPaths(result)
2:     res ← null
3:     for i ← 0 to N do
4:         S ← result.get(i)
5:         k ← result.get(i).size()
6:         if k = maxSize then
7:             res.add(S)
8:         for all s in res do
9:             count ← 0
10:            N ← s.size()
11:            for i ← 0 to N − k + 1 do
12:                if s(i) == result.get(i) then
13:                    for j ← 0 to k do
14:                        if s(i + j) == result.get(j) then
15:                            count + +
16:                        else
17:                            break
18:            if count! = k then
19:                res.add(result.get(i))
20:    return res
```

This algorithm solely operates on the sorted result set obtained from Algorithm 1. Lines 5-7 gets the longest path and stores in k and saves the paths with size k into the final result without any processing. Lines 8-19 uses a variant of sliding window technique to obtain the non-redundant paths and save it to the result. The sliding window algorithm is an optimization algorithm to eliminate unnecessary iterations over a collection and is dependent on the size k. This implies that, fewer iterations are needed when k is large and higher number of iterations are needed when k is small. Line 20 returns the non-redundant paths from the list of paths obtained from Algorithm 1. The runtime complexity of this algorithm is $\mathcal{O}(nm + k)$, where n represents the size of the result set, m represents the size of subpath, and k represents the size of the sliding window.

Theorem 2: Consider any directed acyclic graph $G = (V, E, L)$, and let l_m the label in L based on which the graph traversal is done. The algorithm produces a nonempty subpath set S_p, with edges containing values associated with l_m from result set R_i.

Proof: Let S_p be a maximum-size subset of paths associated with label l_m in R_i. Let $(n_i, ...n_j, ...n_k)$ where $i < j < k$ be the set of nodes connected by edges E_p associated with label l_m in S_p. If the edges connecting $n_i \rightarrow n_j$ and $n_j \rightarrow n_k$ are equal then we are done, $e_{ij} = e_{jk}$, since e_{jk} is a subset of edge set E_p associated with label l_m. If $e_{ij} \neq e_{jk}$, let the edge set $E'_p = E_p - \{e_{jk}\} \cup \{e_{ij}\}$. By substituting e_{ij} for e_{jk} we get $E'_p = E_p$, which shows that e_{ij} and e_{jk} belongs to the same set i.e. having same label. This is true because n_i is a child of n_j and n_k is a child ok n_j. Since $E'_p = E_p$, we conclude that E'_p contains edges with label l_m, and it includes e_{jk}.

6.3. VALUATION FUNCTION

By definition, a valuation function characterized by various parameters returns the value of a state variable depending on the context of the system. In this case, the valuation function returns true if the in-degree for an individual node is greater than 2 for a specific variable.

6.4. SECURITY DOMAINS

Security domains act as a determining factor in the classification of information flow paths in the system based on the trust/integrity level achieved by evaluating the information flow paths. A security domain is considered as a logical partition of the system in which all the paths have an identical valuation function. The security domains are constructed as follows:

- Every component in the system is considered to be in its own security domain.

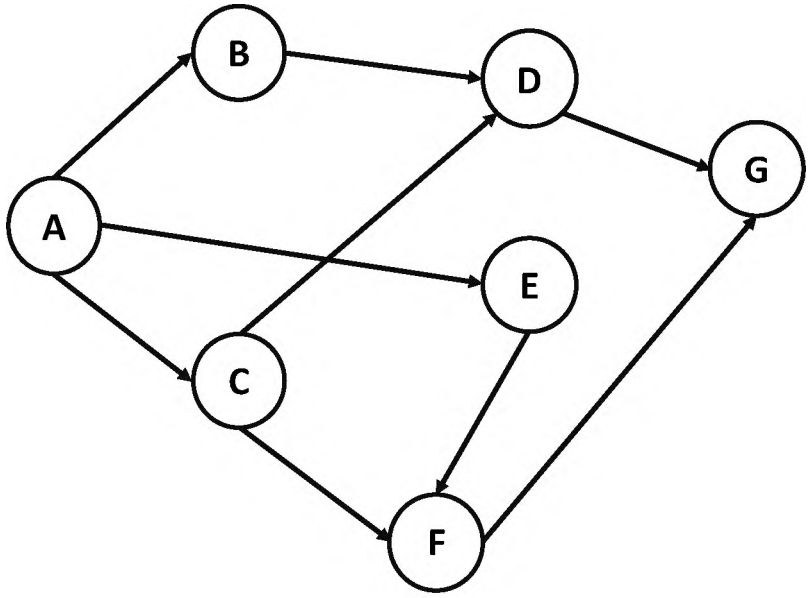


Figure 6.1. Initial Graph

- The process of traversing the graph from one node to another node with respect to a feature combines all the nodes with different security domains into one single domain.
- This process is repeated for all the features which results in multiple security domains (total number of information flow paths).

- If the valuation function exists and the values across all the edges are consistent for a certain label, all the security domains are combined into a single domain called *Secure*. If no valuation function exists, the security domains are combined into a single domain called *Non-Secure*.

To illustrate the above steps, consider a graph G as shown in Figure 6.1. It has seven nodes A - G, which are considered to be in their own security domain. After performing information flow traversal on the graph g with respect to the feature L results in 3 information flow paths, as illustrated in the Figure 6.2. These three information flow paths are considered to be in their own security domains resulting in 3 security domains. The consistency check of the values associated with label L results in two different values x and y. The conflicting values associated with feature L indicates the observer that one of the nodes is faulty or under attack.

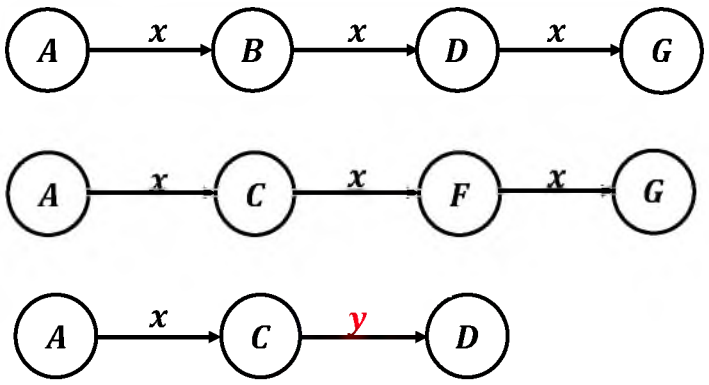


Figure 6.2. Information Flow Paths

The illustrated information flow paths in Figure 6.2 show that node C is sending incorrect data and is said to be faulty. Based on this, the security domains can be further combined into *Secure* and *Non-Secure* domains. This partitioning of the domains helps the observer to identify the problem and make decisions based on the context.

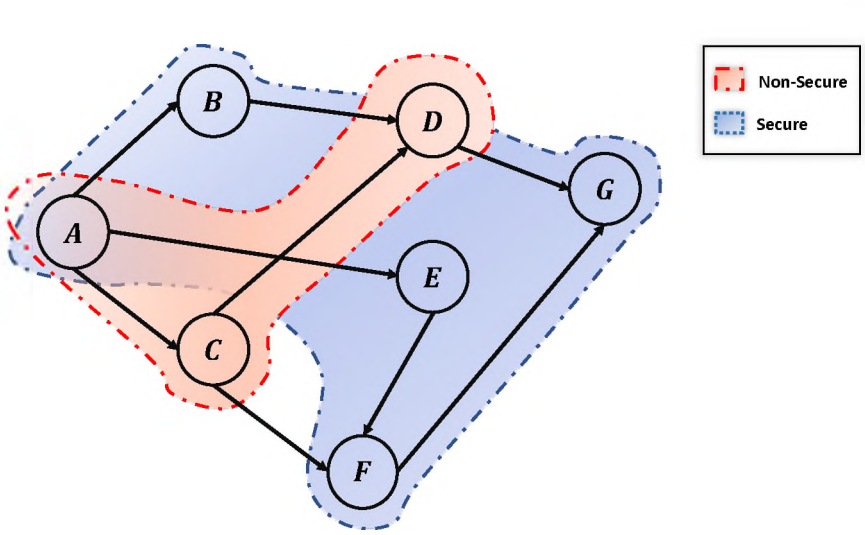


Figure 6.3. Security Domains

Theorem 3: MSDND analysis eventually results in two security domains SD_{Secure} and $SD_{Non-Secure}$ of the directed acyclic graph provided as its input.

Proof: Suppose that MSDND is run on a given DAG, $G = (V, E, L)$, to determine the security partitions with respect to label $l \in L$. It suffices to show that for any pair of distinct vertices $(u, v) \in V$, if security partition SD_{Secure} contains an edge from $u \rightarrow v$ with edge label l represented by $(u \rightarrow v).l$, then $(u \rightarrow v).l = l$. This indicates that the edges connecting the nodes in a particular security domain have consistent label values which ultimately helps in classifying the security partitions. Consider any edge (u, v) explored by MSDND. When this edge is explored, v cannot be added to the security domain unless the edge connecting u to v does not contain label l. Theorem 2 proved that the result set R_i contains non-empty subpath sets S_{p_i}, composed of edges with values associated with a specific label. Therefore, the distinct pair of vertices of a path in SD_{Secure} partition contains an edge from $u \rightarrow v$, such that $(u \rightarrow v).l = l$

7. MSDND ANALYSIS

CPS design is a complex process as one needs to be concerned about physical, cyber, and cyber-physical threats. To alleviate the effects of such attacks, a CPS should be designed to ensure adequate and effective preventative measures are in place at the physical and cyber levels. The physical level measures include physical access restriction to assets, utilizing fault tolerance systems, employee monitoring, and keeping backup copies in remote locations. The cyber level measures include imposing robust cryptography techniques, using certified software, and user/service authentication measures.

The fundamental problems that need to be addressed while building any CPS include:

- Design the most reliable system that meets designated specifications.
- Ensure the system's operation in the best conceivable way to meet specified requirements.

The optimal standard for achieving the problems mentioned above is to define them in terms of the performance measures and financial considerations. The major hurdle is to predict the performance measures of the system before it is developed or operated. This is not straightforward because most designers lack good intuition for complex systems. Additionally, proper computational mechanisms are hard and complex to develop. The data needed is difficult to obtain due to various factors, such as the sensitivity of the data.

CPS security research is structured around the modeling of systems for computing performance measures. The research in this field is based on stochastic processes, approximation methods, non-linear analysis and optimization, statistics, and other modeling techniques. In addition to these measures, there might be cases comprised of unforeseen scenarios such as unexpected failures, problems caused by unreliable machines, causal failures due to old equipment and environmental factors. These factors need to be con-

sidered while coming up with an effective way of handling such scenarios. This section presents the MSDND analysis to address the issues with traditional approaches mentioned earlier. The MSDND analysis helps in identifying the failures and attacks in a CPS during design-time and run-time with the help of graph based analysis by analysing the values associated with the edge labels. MSDND analysis is an effective mechanism that can be implemented at design time and can serve as a potent tool for detecting and mitigating cyber and physical threats. This is achieved by depicting the CPS in terms of a graph network and partitioning them into different security domains based on edge labels. This analysis helps in identifying causal failures such as (i) failures caused by equipment operation outside the actual parameters, (ii) highly reliable equipment failure affecting other equipment, and (iii) failures caused by interacting components operating according to specification.

This section presents real-world case studies demonstrating how MSDND can help identify the cyber-physical vulnerabilities and failures in the avionic systems. Each case study represents either an attack on the system or a failure in the system. The attacks and failures are represented by incorrect values associated with the edge labels. These incorrect values are injected manually to represent an attack or failure. MSDND is applied to these scenarios to identify the vulnerabilities and provide mitigation techniques based on the valuation functions. All the scenarios are divided into two parts: 1) Identifying the compromised system. 2) Applying the graph-based model to identify the information flow paths and use the MSDND model to identify the faulty paths associated with the avionics system. Several aspects of the avionic systems have been evaluated to detect an attack or failure using the proposed methodology, capturing the information flow paths and partitioning them into security domains. Furthermore, the effectiveness of using security domains has been demonstrated to help with design time and real-time attack detection.

7.1. ALTIMETER FAILURE

Theorem: In the case of altimeter compromise, the MSDND model yields deducibility using automated graph-based analysis, thereby allowing critical information flow to the pilot and the controller.

Proof: Consider a scenario in which the barometric altimeter is faulty and is sending incorrect altitude data to the pilot (See Figure 7.1). In this case, the altimeter displays incorrect altitude values and thus making them NonDeducible to the pilot.

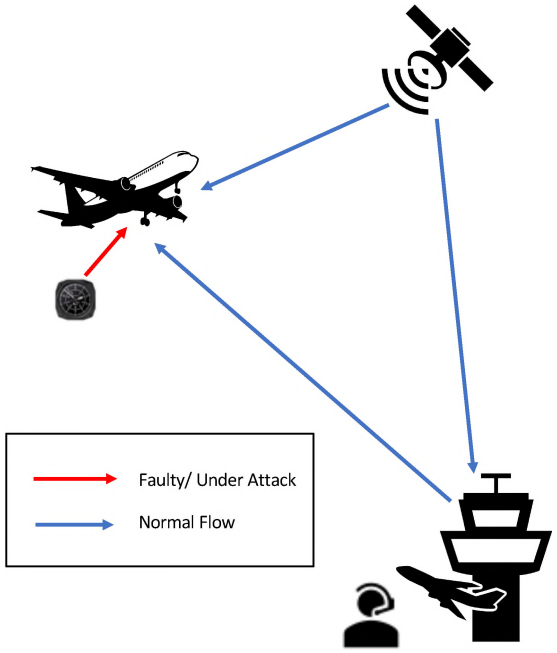


Figure 7.1. ATC Helps to Identify the Faulty Altimeter

The MSDND analysis is performed on graph G with respect to the feature "Altitude", which results in a subgraph, as shown in Figure 7.2. Step 1 results in 129 information flow paths, and the resulting information flow paths set are further reduced by applying step 2. This results in 49 information flow paths by eliminating the subpaths (See Table 7.1). After eliminating the subpaths, step 3 is applied to evaluate the values associated with each label to check for consistency. If there is any inconsistency, the in-degree for each node of the information flow paths is calculated. If a node has an in-degree ≥ 2 indicates that there is

Table 7.1. Scenario 1 - Information Flow Paths

Feature	# Paths	# Independent Paths	MSDND Secure	Nodes
Altitude	129	49	Not ND Secure	6

more than one independent information flow path carrying similar information. If one such node exists, it is considered to have a valuation function, which eventually helps break the NonDeducibility property. If the valuation function exists for a node, the incoming edges are evaluated by backtracking to identify the faulty source. In this scenario, a valuation function exists for node 6, which helps break the NonDeducibility property. Figure 7.3 represents the *Secure* and *Non-Secure* partitions with respect to the feature "Altitude".

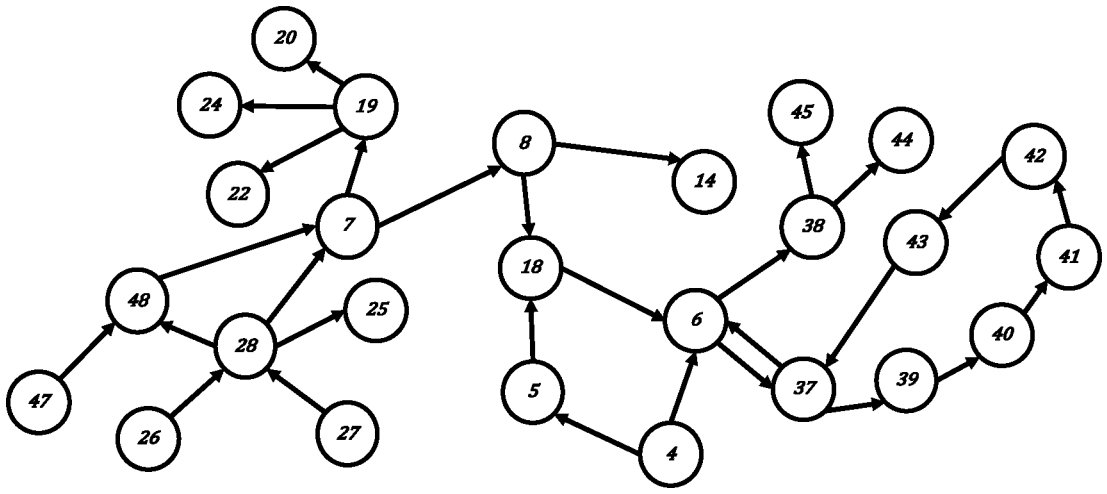


Figure 7.2. Graph - Altitude

The MSDND analysis is applied to verify the correctness of the graph-based MSDND analysis. For this, consider three information flow paths (GPS domain, Altimeter domain, and ATC domain) to verify the correctness. Other information flow paths can also be used to determine if the MSDND property holds.

This scenario considers two security domains, SD^{A} {Altimeter Domain} and SD^{GPS} {GPS Domain}. By combining the valuation functions in SD^{A} and SD^{GPS} with respect to the altitude value from the pilot domain,

$$S_A = \neg\varphi_3 \wedge \neg\varphi_4 \wedge \neg\varphi_5 \wedge \neg\varphi_{28} \wedge \neg\varphi_7 \wedge \neg\varphi_{19} \wedge \neg\varphi_{20} \Rightarrow \nexists V^P_{\sim a} \tag{7.1}$$

Since the information received from the barometric altimeter domain is faulty, the pilot cannot valuate the correctness of the altitude data in that domain.

$$S_{GPS} = \varphi_3 \wedge \varphi_4 \wedge \varphi_5 \wedge \varphi_{28} \wedge \varphi_7 \wedge \varphi_{19} \wedge \varphi_{20} \Rightarrow \nexists V^P_a \tag{7.2}$$

Even though the information received from the GPS domain is not faulty, the correctness of the altitude data cannot be valuated in that domain as the pilot cannot validate the correctness of the data with just two information flow paths.

From Equation 7.1 and Equation 7.2, the pilot can see two different information flow paths which result in different altitude values.

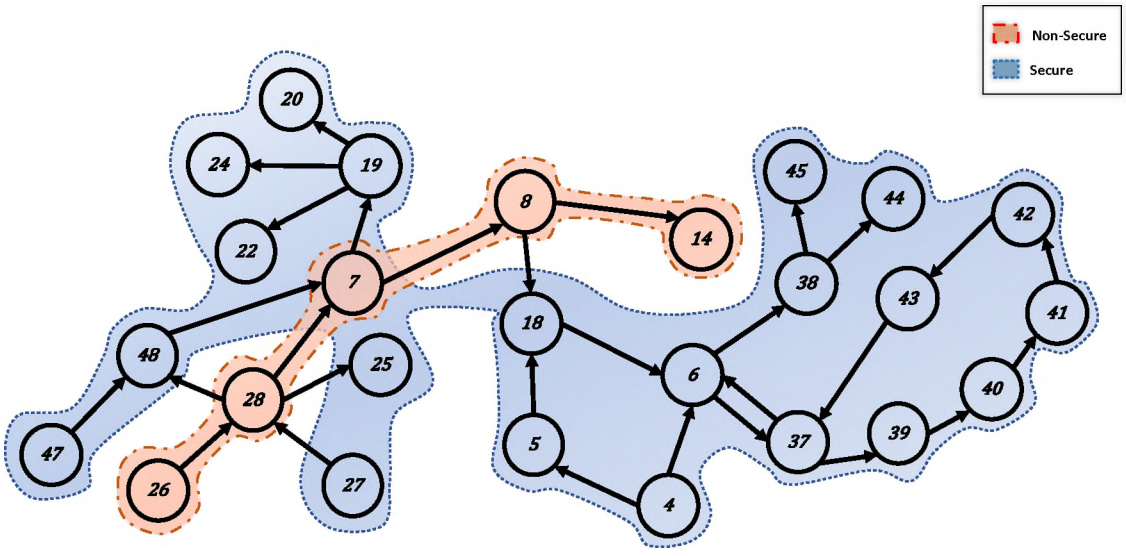


Figure 7.3. Graph - Security Domains

By combining Equation 7.1 and Equation 7.2,

$$MSDND(ES) = \exists w \in W: [w \vdash \Box f(S_A, S_{GPS})] \wedge [w \vDash (\nexists V_{\sim a}^P \wedge \nexists V_a^P] \tag{7.3}$$

Therefore, the pilot cannot deduce that the barometric altimeter is faulty and is sending incorrect altitude data. This situation can be resolved by having an additional information flow path which helps the pilot to resolve the conflict.

$$S_{ATC} = \varphi_{41} \wedge \varphi_{42} \wedge \varphi_{43} \wedge \varphi_{37} \wedge \varphi_6 \wedge \varphi_{28} \wedge \varphi_7 \wedge \varphi_8 \wedge \varphi_{18} \Rightarrow \exists V_a^P \tag{7.4}$$

In this scenario, the additional information flow path is retrieved from the GPS, which is responsible for sending altitude data, as shown in Figure 7.1.

By combining Equation 7.1, Equation 7.2 and Equation 7.11,

$$MSDND(ES) = \exists w \in W: [w \vdash \Box f(S_A, S_{RA}, S_{GPS})] \wedge [w \vDash (\nexists V_{\sim a}^P \wedge \exists V_a^P \wedge \exists V_a^P] \tag{7.5}$$

Hence, the system is *not MSDND secure* to the pilot as he/she can deduce the correct altitude value and thereby resolving the conflict by relying on alternate information flow paths.

7.2. SATELLITE FAILURE

Theorem: In the case of satellite (GNSS) failure, using automated graph-based analysis, the MSDND model yields NonDeducibility, thereby stopping critical information flow to the pilots.

Proof: Consider a scenario in which the GNSS is faulty and is sending incorrect position data to the aircraft. GNSS is responsible for sending the position data to the planes, and the planes communicate with each other. In the case of GNSS failure, the position

information retrieved by the aircraft is incorrect, and the pilots communicate with each other based on this information. This failure could lead to potential mid-air collisions and incorrect decisions by the pilots if they could not identify the source that is sending incorrect data (See Figure 7.4).

Once the flight position is retrieved, the pilots trust the information sent by satellite. If there is another nearby aircraft, the pilots communicate based on the data transmitted by the satellite. To illustrate this in a better way, let Pilot-1 represent the pilot from Plane-1 and Pilot-2 from Plane-2. Pilot-1 and Pilot-2 cannot identify the problem until they are too close, eventually leading to a breakdown in the separation. With the automated MSDND analysis, pilots can check for the consistency of information flow paths and find the source of the erroneous data.

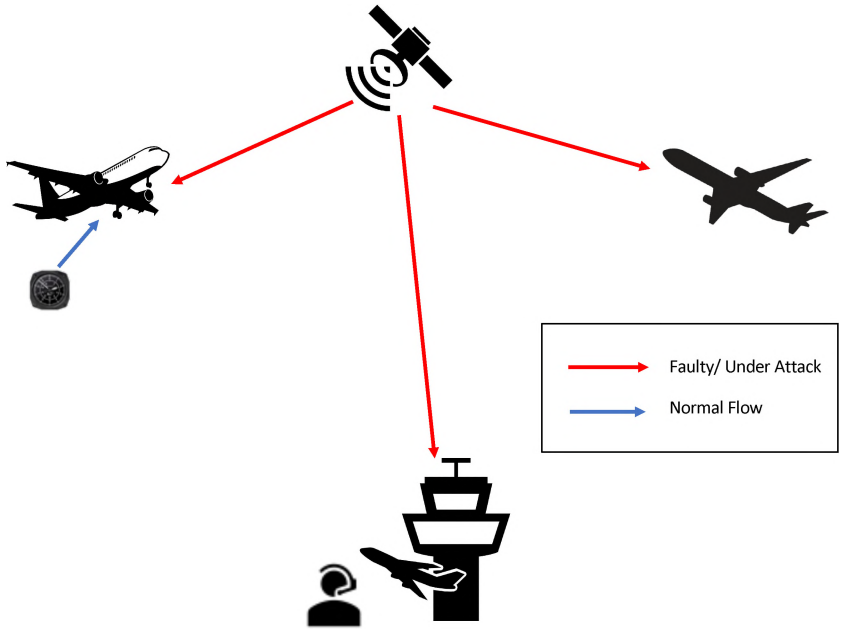


Figure 7.4. Satellite Failure

Figure 7.5 represents the subgraph generated with respect to the label "pos". Once the graph is generated, the sliding window technique with the length equal to the second shortest path from the set is used to eliminate the subgraphs from the set of graphs. Applying Algorithm 2 eliminates the subgraphs to avoid redundancy. In this case, there

Table 7.2. Scenario 2 - Information Flow Paths

Feature	#Paths	#Independent Paths	MSDND Secure	Nodes
Pos	4	3	ND Secure	NA

are no subgraphs, and this step does not result in a reduced set. After eliminating the subgraphs, the value associated with each label is evaluated to check for consistency. If there is any inconsistency, the in-degree for each node in the inconsistent set is calculated. If a node has an in-degree ≥ 2 (indicates that more than one information flow paths are carrying similar information), it is considered to have a valuation function that helps break the NonDeducibility property. If the valuation function exists for a node, the incoming edges are evaluated to identify the faulty source. In this case, none of the nodes has an in-degree value greater than two (See Table 7.2). Therefore, the pilots from both the aircraft cannot deduce that satellite failure is causing the transmission of incorrect information. Figure 7.6 represents the *Secure* and *Non-Secure* partitions with respect to the feature "pos".

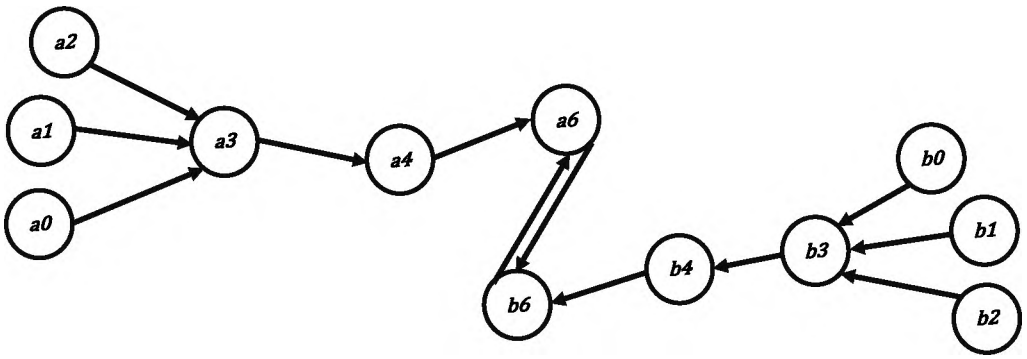


Figure 7.5. Graph - Satellite Failure

Once the flight position is retrieved, Pilot-1 trusts the information sent by Plane-2 and vice-versa. Pilot-1 and Pilot-2 cannot identify the problem until they are too close, which leads to a breakdown in the separation.

By applying MSDND analysis, the correctness of the graph-based MSDND analysis can be verified. In this case, two information flow paths from Pilot-1 Domain, and Pilot-2 domain are considered to verify the correctness.

The two security domains in this scenario are SD^{P1} {Pilot-1 Domain} and SD^{P2} {Pilot-2 Domain}. By combining the valuation functions in SD^{P1} and SD^{P2}.

$$S_{p1} = \neg\varphi_{a0} \wedge \neg\varphi_{a1} \wedge \neg\varphi_{a2} \wedge \neg\varphi_{a3} \wedge \neg\varphi_{a4} \Rightarrow \nexists V^{P1}_{\sim pos} \tag{7.6}$$

Since the information received from the Pilot-2 domain is faulty, Pilot-1 cannot evaluate the correctness of the position data in that domain.

$$S_{p2} = \neg\varphi_{b0} \wedge \neg\varphi_{b1} \wedge \neg\varphi_{b2} \wedge \neg\varphi_{b3} \wedge \neg\varphi_{b4} \Rightarrow \nexists V^{P2}_{\sim pos} \tag{7.7}$$

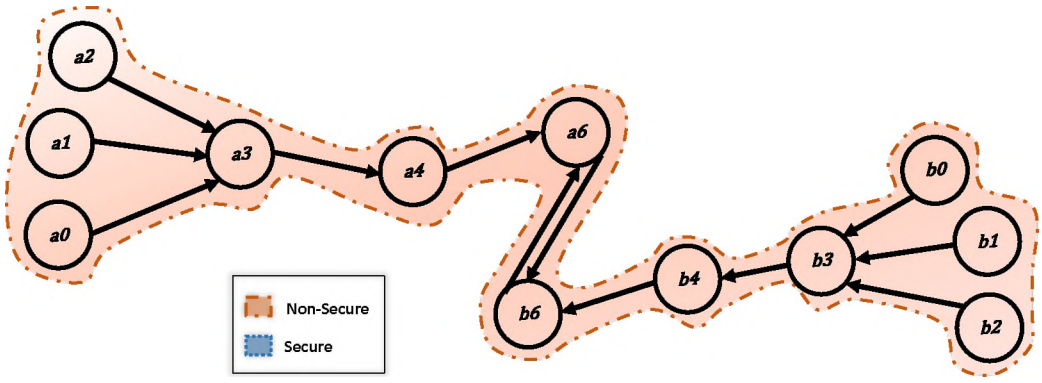


Figure 7.6. Graph - Security Domains

Since the information received from the Pilot-1 domain is faulty, Pilot-2 cannot valuate the correctness of the position data in that domain.

By combining Equation 7.9 and Equation 7.10,

$$MSDND(ES) = \exists w \in W: [w \vdash \Box f(S_{p1}, S_{p2})] \wedge [w \vDash (\nexists V^{P1}_{\sim pos} \wedge \nexists V^{P2}_{\sim pos}] \tag{7.8}$$

Therefore, the pilots from both the aircraft cannot deduce that satellite failure is causing the transmission of incorrect information.

Hence, the system is *MSDND secure* to the pilots as they cannot deduce the actual true position of the planes.

7.3. PITOT STATIC SYSTEM (PSS) FAILURE

Theorem: In the case of the PSS's primary static source failure, using automated graph-based analysis, the MSDND model yields deducibility, thereby allowing critical information flow to the pilots.

Proof: Consider a scenario in which the primary static source is blocked as shown in Figure 7.7, and an alternate static source is used to provide static pressure. When this is used as a primary source for pressure, the altimeter projects a higher altitude reading, ASI displays higher airspeed than the actual airspeed, and VSI reports a momentary climb. In such cases, pilots are advised to rely on the flight manual or pilot's operating handbook to calculate the error.

With the automated MSDND analysis, pilots can check for the consistency of information flow paths and find the source of the erroneous data without relying on manuals or handbooks. As MSDND captures all the information flow paths and performs checks

for consistency, pilots can be notified when there is a discrepancy. If the valuation function exists for airspeed and altitude, pilots can rely on other sources rather than computing the error manually.

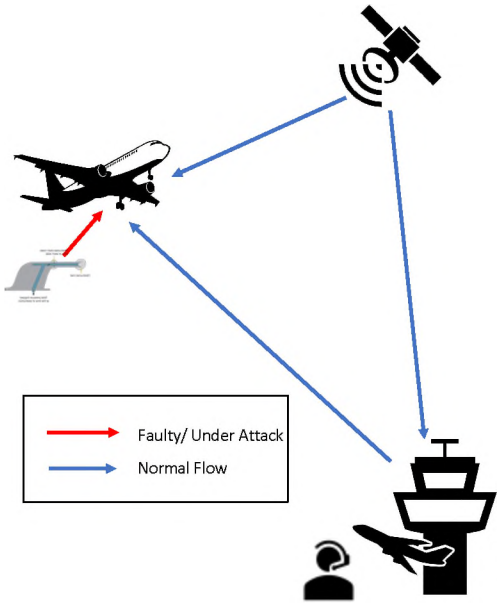


Figure 7.7. Pitot Static System Failure

Figure 7.8 represents the subgraph generated with respect to the labels *altitude* and *airspeed*. Applying step 1 and step 2 of MSDND analysis results in 138 and 54 information flow paths, respectively (See Table 7.3). Step 3 results in an in-degree greater than two for nodes ADS-B Transponder and Air Data Computer. These nodes help break the NonDeducibility property and ensure that the pilot need not rely on inconsistent data. Figure 7.9 represents the *Secure* and *Non-Secure* partitions with respect to the feature *altitude* and *airspeed*.

The MSDND analysis is applied to verify the correctness of the graph-based MSDND analysis. In this case, three information flow paths from the PSS, radio altimeter (RA), and GPS domain are considered to verify the correctness.

Table 7.3. Scenario 3 - Information Flow Paths

Feature	#Paths	#Independent Paths	MSDND Secure	Nodes
Altitude	129	49	Not ND Secure	6, 7, 18, 28, 37, 48
Airspeed	9	5	ND Secure	18

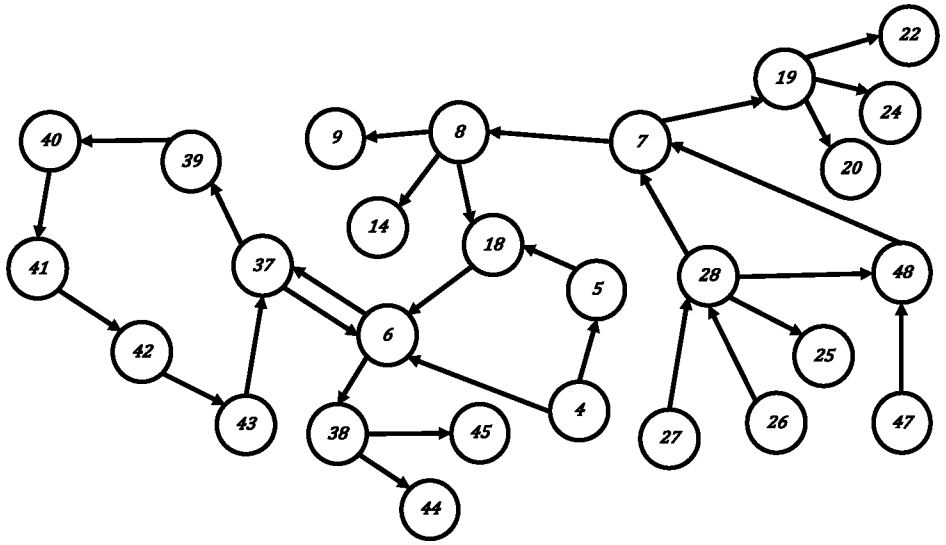


Figure 7.8. Graph - Altitude & Airspeed

The two security domains in this scenario are SD^{PSS} {pitot static system Domain} and SD^{RA} {radio altimeter}. By combining the valuation functions in SD^{PSS} and SD^{RA},

$$S_{PSS} = \neg\varphi_{26} \wedge \neg\varphi_{28} \wedge \neg\varphi_{7} \wedge \neg\varphi_{8} \wedge \neg\varphi_{14} \Rightarrow \nexists V^{P}_{\sim a} \tag{7.9}$$

Since the information received from the Pilot-2 domain is faulty, Pilot-1 cannot evaluate the correctness of the position data in that domain.

$$S_{RA} = \varphi_{47} \wedge \varphi_{48} \wedge \varphi_{7} \wedge \varphi_{19} \wedge \varphi_{22} \Rightarrow \nexists V^{P}_{a} \tag{7.10}$$

Therefore, pilots cannot deduce that the pitot-static system is faulty and is sending incorrect altitude data. This situation can be resolved by having an additional information flow path that helps the pilots resolve the conflict.

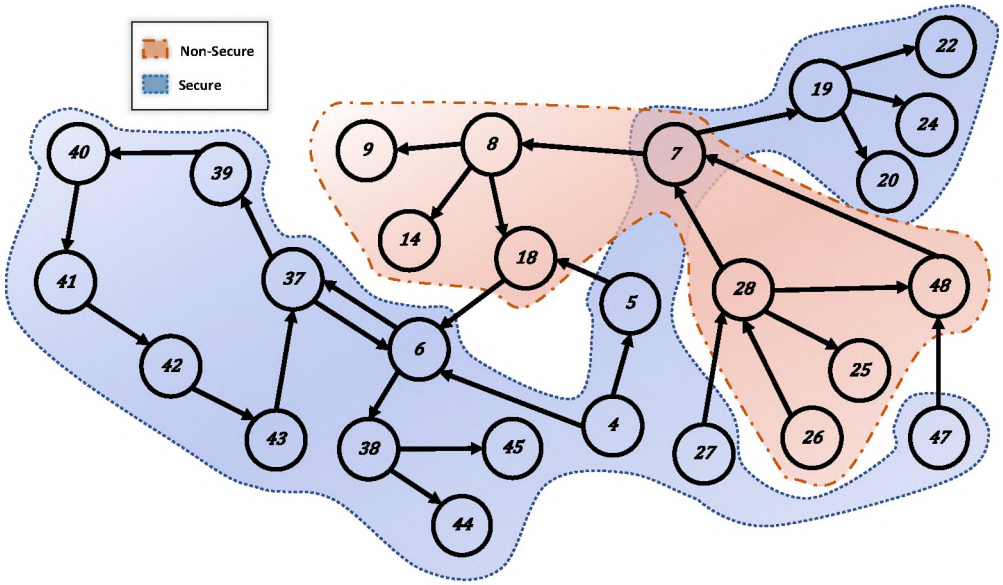


Figure 7.9. Graph - Security Domains

$$S_{GPS} = \varphi_3 \wedge \varphi_4 \wedge \varphi_5 \wedge \varphi_{28} \wedge \varphi_7 \wedge \varphi_{19} \wedge \varphi_{20} \Rightarrow \exists V_a^P \tag{7.11}$$

In this scenario, the additional information flow path is retrieved from the ATC controller, which is responsible for sending altitude data, as shown in Figure 7.7.

By combining Equation 7.1, Equation 7.2 and Equation 7.11,

$$\begin{aligned} MSDND(ES) = \exists w \in W: [w \vdash \Box f(S_A, S_{GPS}, S_{ATC})] \\ \wedge [w \vDash (\nexists V_{\sim a}^P \wedge \exists V_a^P \wedge \exists V_a^P] \end{aligned} \tag{7.12}$$

Hence, the system is *not MSDND secure* to the pilots as they can deduce the correct altitude value and thereby resolving the conflict by relying on alternate information flow paths. Therefore, the pilots can deduce that the pitot-static system is faulty.

7.4. PRIMARY FLIGHT DISPLAY FAILURE

Theorem: In the case of primary flight display's (PFD) failure, the MSDND model yields deducibility, thereby allowing critical information flow to the pilots.

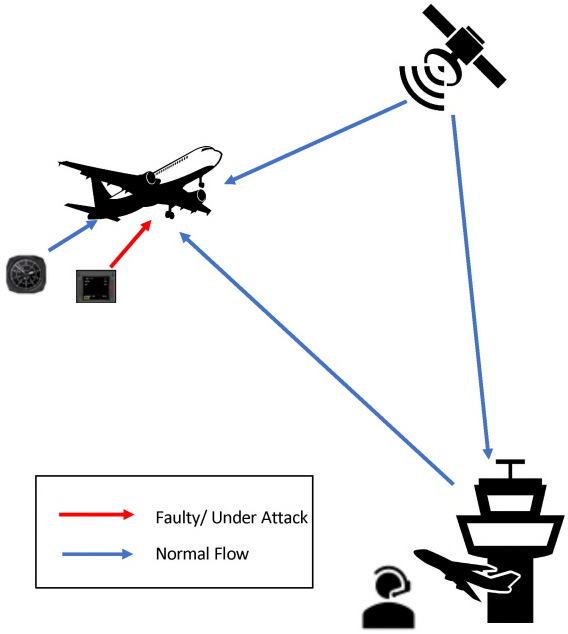


Figure 7.10. Primary Flight Display Failure

Proof: Consider a scenario in which the PFD functions incorrectly as shown in Figure 7.10. PFD serves as the primary reference for flight information for pilots. This system gathers information from various sources and displays it in order to reduce the pilot's workload and increase situational awareness. As data from various sources is displayed on PFD, any failure in the PFD could lead to projecting incorrect data or no data at all.

With the automated MSDND analysis, pilots can check for the consistency of information flow paths and find the source of the erroneous data in addition to the Multi-Function Display (MFD) present in the cockpit. As MSDND captures all the information flow paths and performs checks for consistency, pilots can be notified when there is a discrepancy. If a valuation function exists for any of the data, pilots can rely on other sources rather than computing the error manually.

Table 7.4. Scenario 4 - Information Flow Paths

Feature	#Paths	#Independent Paths	MSDND Secure	Nodes
Altitude	129	49	Not ND Secure	6, 7, 18, 28, 37, 48
Air Density	8	4	Not ND Secure	18
Airspeed	9	5	Not ND Secure	18
Heading	19	8	Not ND Secure	18, 31
OAT	5	3	ND Secure	NA
TAT	5	3	ND Secure	NA
Turn	12	3	ND Secure	NA
Vertical Speed	13	6	Not ND Secure	28

Figure 7.11 represents the subgraph generated with respect to the labels *Air Speed, Turn, Heading, Altitude, Vertical Speed, Attitude, AirSpeed, Air Density, Outside Air Temperature* and *Total Air Temperature*.

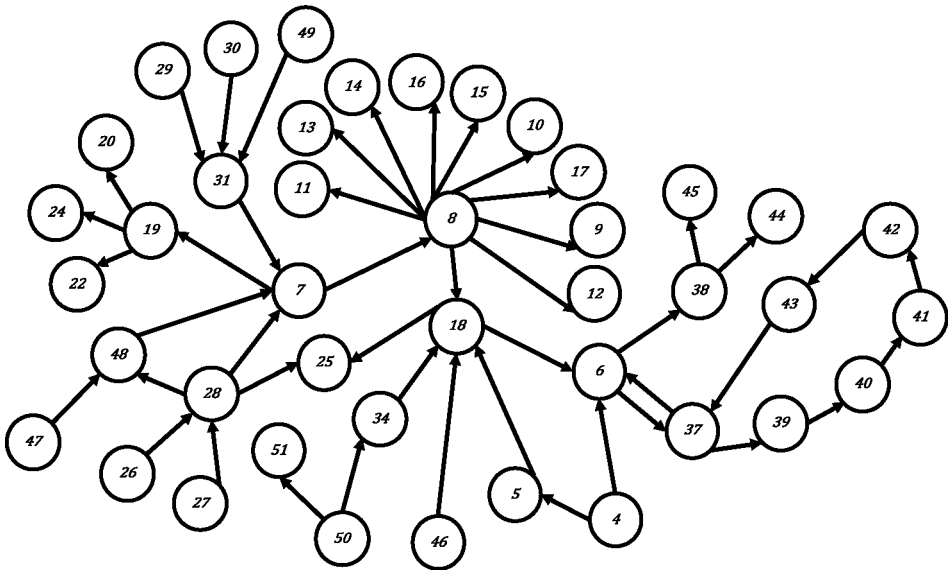


Figure 7.11. Graph - Air Speed, Turn, Heading, Altitude, Vertical Speed, Attitude, Air Speed, Air Density, Outside Air Temperature & Total Air Temperature

Applying step 1 and step 2 of MSDND analysis results in 186 and 81 information flow paths, respectively (See Table 7.4). Step 3 results in an in-degree greater than two for the nodes ADS-B Transponder and Air Data Computer. These nodes help break the NonDeducibility property and thus ensuring that the pilot need not rely on inconsistent data.

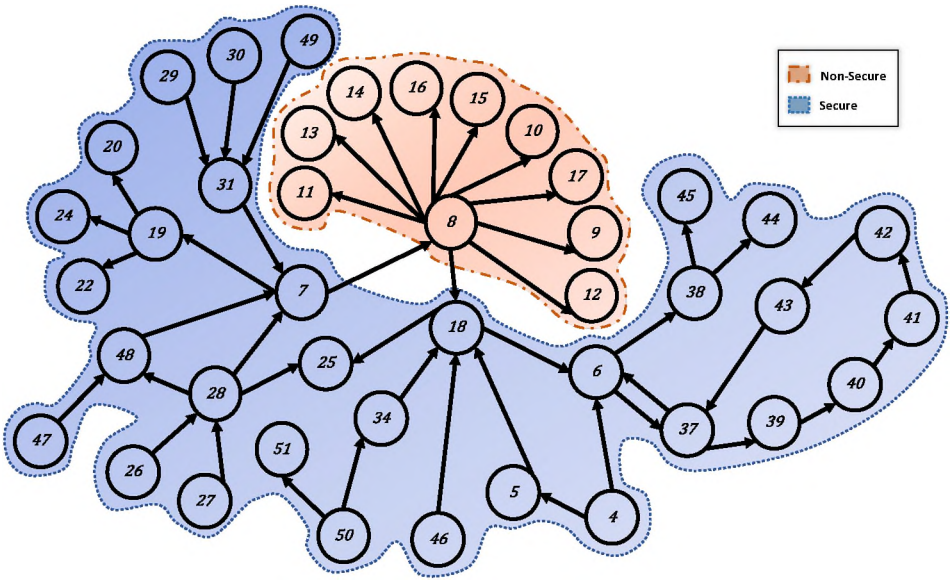


Figure 7.12. Graph - Security Domains

Finally, step 5 is applied to partition the subgraphs based on the consistency of the information flow paths into *Secure* and *Non-Secure* domains. Figure 7.12 represents the *Secure* and *Non-Secure* partitions with respect to attitude and heading.

Hence, the system is *not MSDND secure* to the pilots as they can deduce the faulty source and thereby relying on alternate information flow paths to operate the aircraft.

7.5. MAGNETOMETER FAILURE

Theorem: In the case of magnetometer failure, the MSDND model yields deducibility, thereby allowing critical information flow to the pilots.

Table 7.5. Scenario 5 - Information Flow Paths

Feature	#Paths	#Independent Paths	MSDND Secure	Nodes
Attitude	14	6	Not ND Secure	6, 18
Heading	19	8	Not ND Secure	6, 18

Proof: Consider a scenario in which the magnetometer is faulty. In aeronautics, the magnetometer can be used to measure the geomagnetic field vector information of the position of the aircraft body, such as airplanes and satellites. According to the reference model for the earth's magnetic field and local magnetic field, the angle information of a certain precision can be obtained through an algorithm. Therefore, the magnetometer is widely used in aircraft attitude determination systems, especially in microsatellites, such as nanosatellites and picosatellites, etc. Since this is one of the crucial instruments to determine the attitude, ensuring the normal function of this is of great importance.

Table 7.5, illustrates the total number of information flow paths and valuation functions associated with each label.

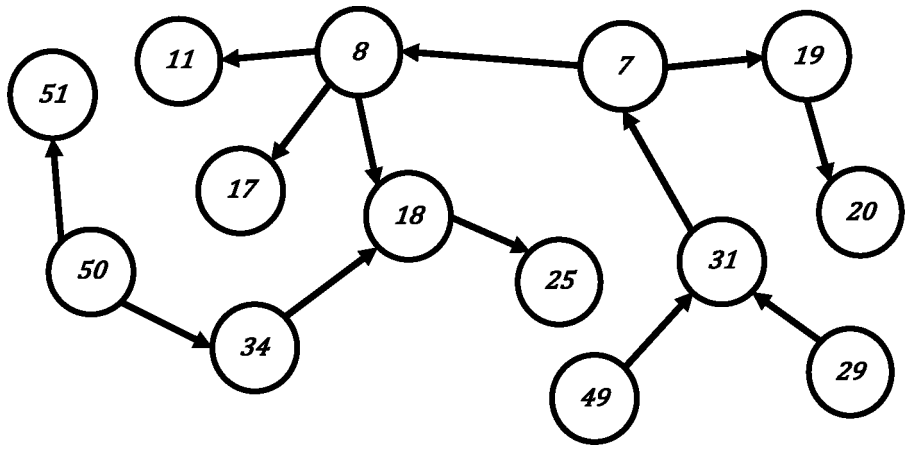


Figure 7.13. Graph - Attitude & Heading

With the automated MSDND analysis, pilots can check for the consistency of information flow paths and find the source of the erroneous data in case of an attack or a failure. As MSDND captures all the information flow paths and performs checks for consistency, pilots can be notified when there is a discrepancy. If a valuation function exists for any of the data, pilots can rely on other sources rather than computing the error manually.

Figure 7.13 represents the subgraph generated with respect to the labels *Attitude* and *Heading*. Applying step 1 and step 2 of MSDND analysis results in 33 and 14 information flow paths, respectively. Step 3 results in an in-degree greater than two for nodes Flight Management System (FMS) and Attitude and Heading Reference System (AHRS) (See Table 7.5). These nodes help break the NonDeducibility property and thus ensuring that the pilot need not rely on inconsistent data.

Finally, step 5 is applied to partition the subgraphs based on the consistency of the information flow paths into $Secure$ and $Non-Secure$ domains. Figure 7.14 represents the $Secure$ and $Non-Secure$ partitions with respect to attitude and heading.

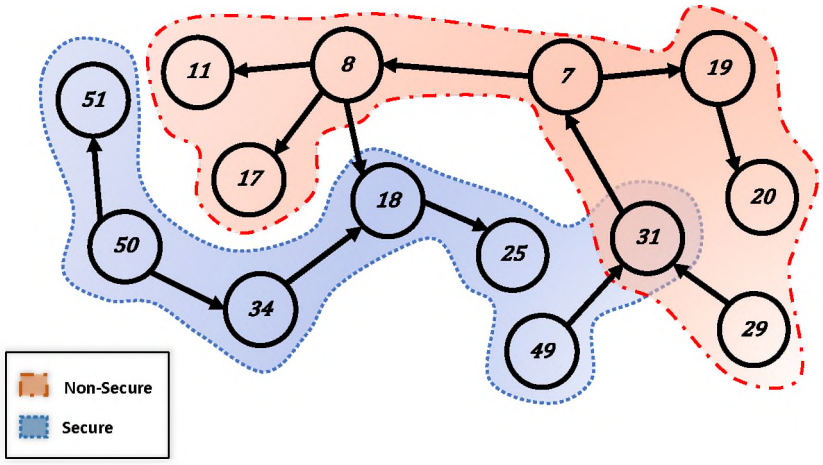


Figure 7.14. Graph - Security Domains

Table 7.6. Scenario 6 - Information Flow Paths

Feature	#Paths	#Independent Paths	MSDND Secure	Nodes
Latitude	64	25	Not ND Secure	6, 18
Longitude	64	25	Not ND Secure	6, 18

7.6. RF INTERFERENCE

Theorem: In the case of RF interference, the MSDND model yields deducibility, thereby allowing critical information flow to the pilots.

Proof: Radio Frequency (RF) Interference is the term used to describe a range of situations in which transmissions such as unwanted other than those from authorized users of an RTF frequency interfere with radio reception.

Table 7.6, illustrates the total number of information flow paths and valuation functions associated with each label.

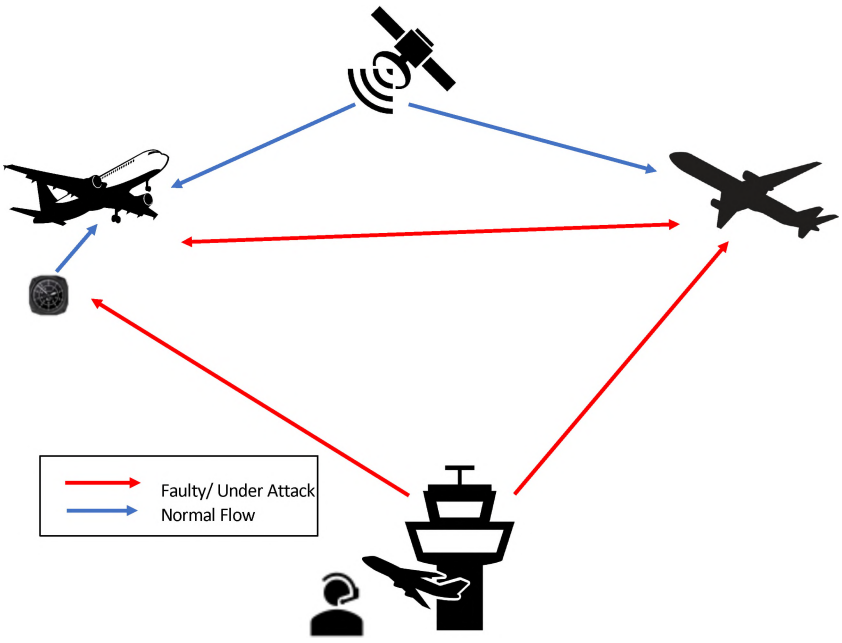


Figure 7.15. ATC Disconnect

Radio interference usually comes from commercial stations on the ground. Some of the key contributing factors for RF interference are weather, atmospheric conditions, malicious and unauthorized transmissions. Interference can make communication difficult or even impossible, resulting in loss of communication. This interference eventually results in an increase in pilot workload and ATC workload and causes callsign confusion. Some of the known problems associated with RF interference are ATC disconnect, autopilot disconnect during the cruise, blank displays, uncommanded rolls or turns, FMS or autopilot standby, inoperative altimeter.

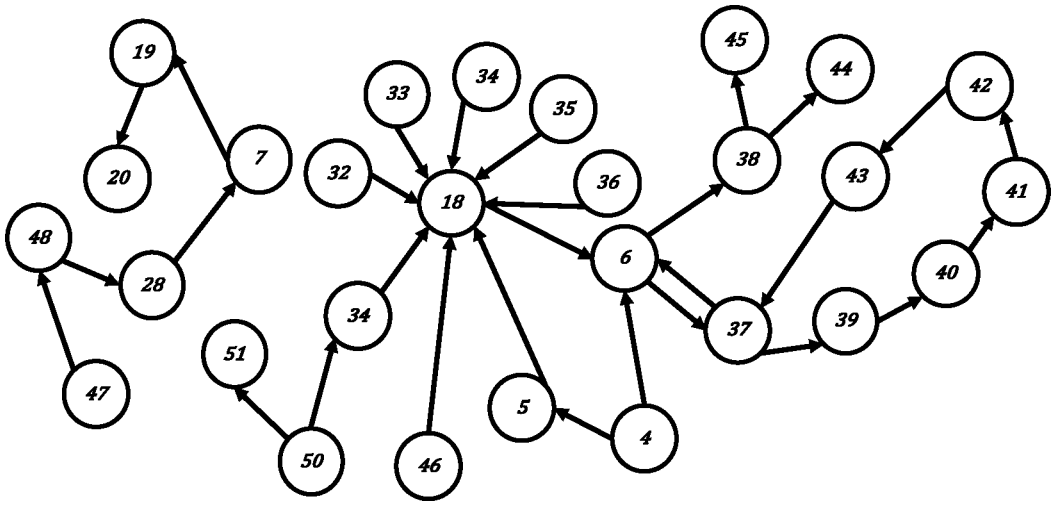


Figure 7.16. Graph - Latitude & Longitude

Let us assume that the ATC connection is lost with the pilots due to RF interference (See Figure 7.15). The MSDND analysis is applied to check whether the cause of the problem can be identified. With the MSDND analysis, pilots can check for the consistency of information flow paths and find the source of the erroneous data or stop relying on inoperative instruments in case of an attack or a failure. As MSDND captures all the information flow paths and performs checks for consistency, pilots can be notified when

there is a discrepancy or blank displays. If a valuation function exists for any of the data such as latitude, longitude, or altimeter, pilots can rely on other sources rather than relying on handbooks or operation manuals.

Figure 7.16 represents the subgraph generated with respect to the labels *latitude* and *longitude*. Applying step 1 and step 2 of MSDND analysis results in 64 and 25 information flow paths, respectively (See Table 7.6). Step 3 results in an in-degree greater than two for nodes flight management system and ADS-B transponder. These nodes help in breaking the NonDeducibility property and thus ensuring that the pilot need not rely on inconsistent data.

Finally, step 5 is applied to partition the subgraphs based on the consistency of the information flow paths into *Secure* and *Non-Secure* domains. Figure 7.17 represents the *Secure* and *Non-Secure* partitions with respect to latitude and longitude.

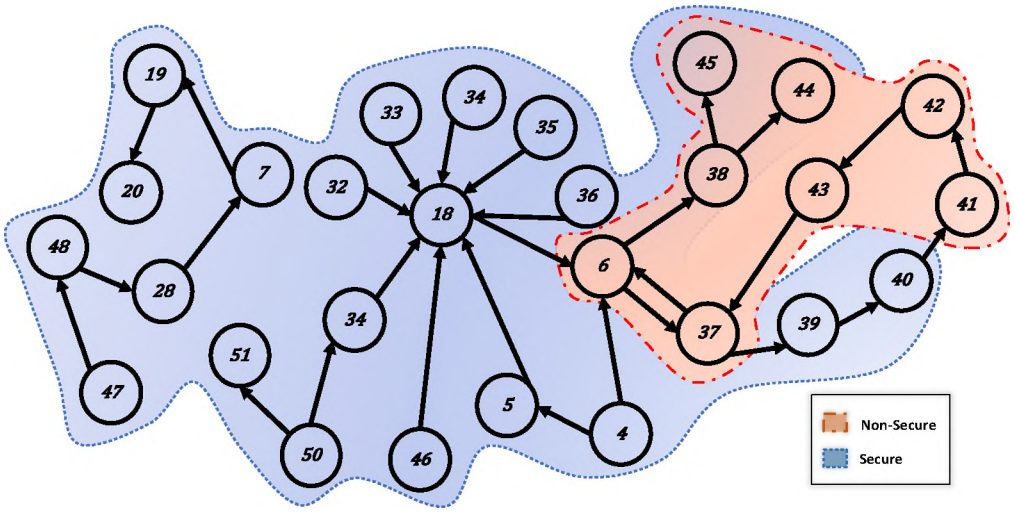


Figure 7.17. Graph - Secure and Non-Secure

7.7. UNRELIABLE AIRSPEED INDICATIONS

Theorem: In the case of unreliable airspeed indications, the MSDND model yields NonDeducibility, thereby stopping critical information flow to the pilots.

Proof: A failure to promptly recognize and respond to erroneous flight instrument indications can result in loss of control. To handle such situations, pilots should be aware of the PSS's functioning and an understanding of the types of erroneous indications that can occur. The PSS will help the pilots realize that there is a problem and follow procedures to establish and maintain the aircraft in a safe condition by referencing the aircraft attitude, the thrust setting, and altitude as verified from at least two similar displays from independent data sources. There is an enormous manual work involved in this process, and this might lead to heavily relying on manuals to come up with an alternative solution.

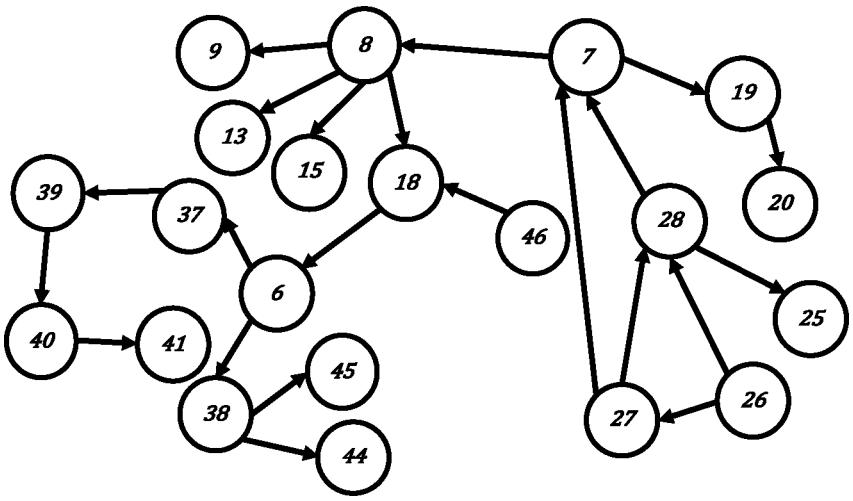


Figure 7.18. Graph - Vertical Rate, Airspeed, Vertical Speed and Air Density

This situation is quite challenging as the pilots may only become aware of the problem when the aircraft has adopted an unusual pitch attitude. This situation gets further complicated, especially when the aircraft has entered a stall. The stall is a condition in which an aircraft cannot produce the required lift for regular operation as it exceeded its

Table 7.7. Scenario 7 - Information Flow Paths

Feature	#Paths	#Independent Paths	MSDND Secure	Nodes
Vertical Rate	13	4	ND Secure	NA
Airspeed	9	5	Not ND Secure	18
Vertical Speed	13	6	Not ND Secure	28
Air Density	8	4	not ND Secure	18

given critical angle of attack. In simple terms, a stall is defined as the aerodynamic loss of lift that occurs when an airplane's wing exceeds its critical angle of attack. Furthermore, in the event of unreliable speed, autopilot, auto thrust, and flight directors can all contribute to loss of control.

With the MSDND analysis, pilots can check for the consistency of information flow paths and find the source of the erroneous data or stop relying on inoperative instruments in case of an attack or a failure. As MSDND captures all the information flow paths and performs checks for consistency, pilots can be notified when there is a discrepancy or blank displays. If a valuation function exists for any of the data such as altimeter, autopilot mode, turn, and heading, the pilot can rely on other sources rather than relying on handbooks or operation manuals.

Figure 7.18 represents the subgraph generated with respect to the labels *vertical rate, airspeed, vertical speed* and *air density*. Applying step 1 and step 2 of MSDND analysis results in 43 and 19 information flow paths, respectively. Step 3 results in an in-degree greater than two for nodes flight management system and ADC. These nodes help in breaking the NonDeducibility property and thus ensuring that the pilots need not rely on inconsistent data.

Table 7.7, illustrates the total number of information flow paths and valuation functions associated with each label.

Finally, step 5 is applied to partition the subgraphs based on the consistency of the information flow paths into $Secure$ and $Non-Secure$ domains. Figure 7.19 represents the $Secure$ and $Non-Secure$ partitions with respect to attitude and heading.

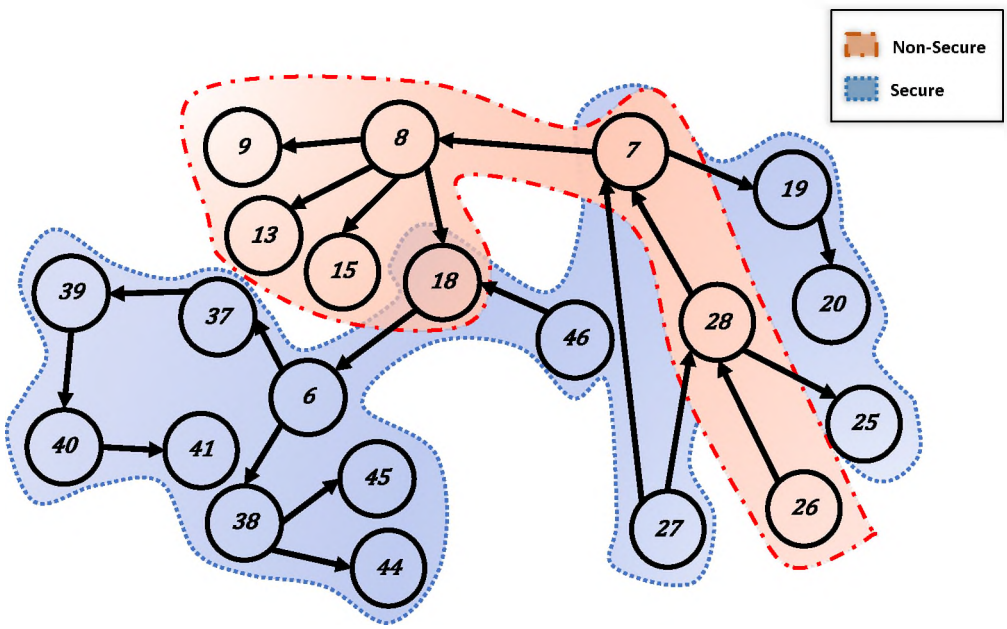


Figure 7.19. Graph - Secure and Non-Secure

7.8. SUMMARY

This section presents a summary of the MSDND analysis performed on all the scenarios. MSDND analysis is performed on each feature to identify the independent information flow paths and the nodes that have a valuation function. This is essential to identify the faulty components in the system or attacks. This analysis can be used to identify design failures as well as can aid the pilots in identifying anomalous patterns in case of an attack or failure in real-time. During design-time, this analysis can be used to identify the vulnerabilities and propose possible mitigation measures. Table 7.8, illustrates the total number of information flow paths and valuation functions associated with each label.

Figure 7.20 shows the information flow paths and NonDeducibility. The MSDND analysis is performed on the data collected by referring to the flight manuals, pilot handbooks, and from ADS-B exchange platform. Based on this analysis, it is evident that performing consistency checks on the values associated with the edge labels of the information flow paths gives an adequate understanding of the system's vulnerabilities. Additionally, Figure 7.20 shows that the valuation function evaluating to true is higher for a feature with a higher number of independent information flow paths that represent information flow from different components in a CPS. In some cases, the valuation function is evaluated to false even in the presence of a higher number of independent information flow paths. This result indicates that there is not enough redundancy for that particular feature, and appropriate measures can be taken during design-time to improve reliability in case of a failure and thereby enhance safety. Hence, MSDND can be used as an effective design-time analysis tool to identify vulnerabilities in the system and propose possible mitigation measures.

Table 7.8. Information Flow Paths of the Components in Avionic Systems

Feature	#Paths	#Independent Paths	MSDND Secure	Nodes
Altitude	129	49	Not ND Secure	6, 7, 18, 28, 37, 48
Aircraft Course	2	2	ND Secure	NA
Aircraft ID	13	4	ND Secure	NA
Air Density	8	4	Not ND Secure	18
Airspeed	9	5	Not ND Secure	18
Angular Rate	2	1	ND Secure	NA
Distance	13	5	ND Secure	NA
Dynamic Pressure	1	1	ND Secure	NA
ETBW	3	3	ND Secure	NA
FFTR	3	3	ND Secure	NA
Fuel Consumed	3	3	ND Secure	NA
Fuel Level	1	1	ND Secure	NA
Ground Speed	28	8	Not ND Secure	6
Heading	19	8	Not ND Secure	18, 31
IAS	4	2	ND Secure	NA
Latitude & Longitude	64	25	Not ND Secure	6, 18, 37
Mach Number	4	2	ND Secure	NA
OAT	5	3	ND Secure	NA
Pos	4	3	Not ND Secure	3
RA	2	2	ND Secure	NA
Static Pressure	2	2	ND Secure	NA
TA	2	2	ND Secure	NA
TAT	5	3	ND Secure	NA
Time	3	3	Not ND Secure	3, 6
Track Angle	11	3	ND Secure	NA
True Airspeed	4	2	ND Secure	NA
Turn	12	3	ND Secure	NA
Velocity	4	2	ND Secure	NA
Vertical Rate	13	4	ND Secure	NA
Vertical Speed	13	6	Not ND Secure	28

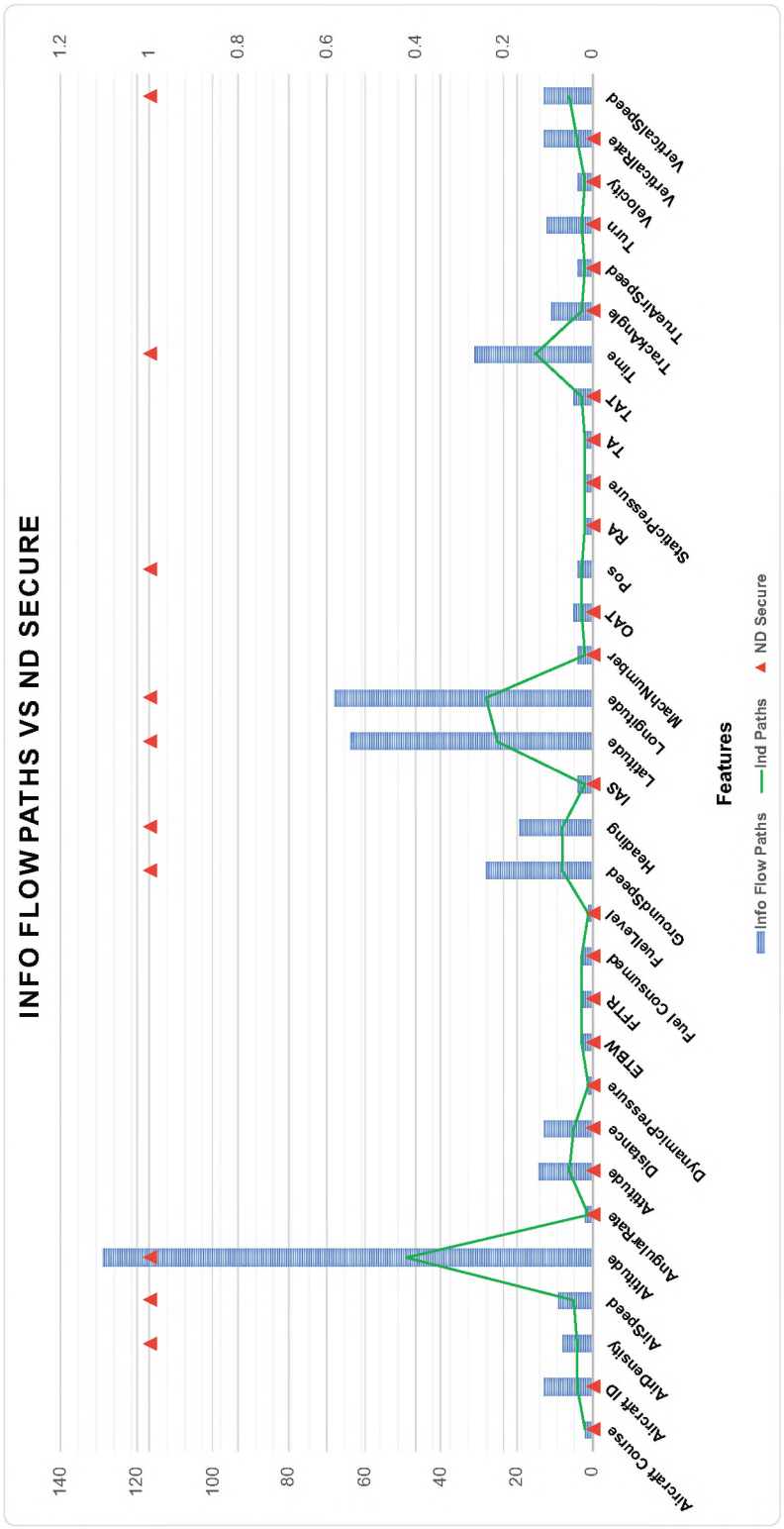


Figure 7.20. Information Flow Paths vs NonDeducibility (*The x-axis represents components in an aircraft. The y-axis represents information flow paths, the solid line indicates independent information flow paths, and the triangle represents whether the component is MSDND secure in case of an attack or failure.*)

8. MPENN - MESSAGE PASSING EDGE CONVOLUTIONS NEURAL NETWORK

8.1. GRAPH NEURAL NETWORKS (GNN)

Graph analysis using convolutional neural networks (CNNs) has become widely popular due to the ability to capture rich relational information among entities using structural graphs. CNNs are prevalent in fields such as computer vision, speech recognition, or natural language processing (Simonovsky and Komodakis, 2017; Wu *et al.*, 2020). Anomaly detection is another important problem that has been well-studied within distinct research areas and application ranges. The anomalies or outliers represent a deviation in the data. According to (Hawkins, 1980), an outlier or anomaly is defined as an observation that "deviates so significantly from other observations as to arouse suspicion that it was generated by a different mechanism." In all these fields, the underlying data representation is in the form of a layer structure. In contrast, many other structures such as physical systems, social networks, computational chemistry, and 3D-modeling have the data represented in non-euclidean domains, which is easier to express in terms of graphs.

Nevertheless, this transition from grids to graphs of CNNs is complex and has recently gained interest among researchers. Researchers have introduced GNNs that are analogous to hierarchical CNN-like architecture to handle graph structures. Modeling complex systems such as CPSs requires a model to learn from graph data. Graph structured data provides scope for prediction tasks on a variety of real-world problems. This work is an attempt to apply graph neural networks to critical infrastructures, to partition the security space of a specific CPS into different domains. A graph classification algorithm using GNNs is introduced in this section to achieve partitioning functionality. Performing such

an analysis using GNN is still in its initial stages due to the unavailability of large datasets of CPS's attack data. Since there is limited data available for a CPS, datasets from the chemical domain are used for analysis.

The datasets used in this section has the following structure:

- Each chemical compound has a set of subgraphs with edge and node features. This is similar to the subpaths associated with the particular feature discussed in Section 7. For example, the information flow paths associated with the feature *altitude* while traversing the graph.

- Each subgraph is associated with a label representing a certain property of the chemical compound. This is akin to the domain label associated with information flow paths i.e. Secure and Non-Secure.

Note: In the remainder of this section, the terms "class label" and "classes" are used to represent the *Secure* and *Non-Secure* security domain partitions.

A few anomalies have been injected into the data that represent anomalies in a CPS to perform the classification of the subgraphs into Secure and Non-Secure classes based on domain partitioning. For example, consider the case of a compromised altimeter which presents incorrect altitude readings. The anomalies, in this case, are represented by incorrect altitude interpretations that differ from the ones retrieved from other components in an aircraft, such as the GPS. Based on the actual and anomalous data, the proposed GNN model classifies them into one of the two associated labels (in case of CPS, it is *Secure* and *Non-Secure*) in the dataset. To represent these sort of instances, the edge features of a few instances in the chemical compound datasets has been tweaked so that one class represents data that belongs to *Secure* domain whereas the other class represents anomalous data that belongs to *Non-Secure* domain.

The anomalies in chemical compounds originate from thermodynamic and structural alterations (de Oliveira *et al.*, 2006; Greer and Moldover, 1981). Structural anomalies include distortions, phase changes, electronic configurations. Thermodynamic anomalies include the effect of pressure and temperature. The analogous structural and thermodynamic anomalies in a CPS include compromised control systems (malware and sending false control signals), manipulated communication traffic, compromised sensors, and actuators due to an attack or failure, incorrect information transmission due to wear and tear of physical components. In the aircraft example, in case of an attack or failure, anomalies can be in the form of incorrect sensor readings or display values. This similarity between the anomalies in CPS and chemical compounds provided a strong basis to utilize the datasets from the chemical domain to perform the security domain classification. In this work, the primary goal is to demonstrate the GNN models' effectiveness in automating the process of partitioning the CPS's security domains. This supervised learning can be used in real-time with respect to a CPS to identify the vulnerabilities and partition them into security domains.

In this dissertation, a Message Passing Neural Network (MPNN) (Gilmer *et al.*, 2017), a general framework for supervised learning on graphs, has been used along with edge convolutions (Simonovsky and Komodakis, 2017). Though there is increased research related to the MPNN framework, there is a need for new variations that can fit well for applications such as partitioning the security domain in a CPS. In this work, a neural network architecture that accepts graphs of arbitrary structure has been introduced. Given a dataset containing graphs in the form of (G, y) where G represents a graph and y represents its class, the model reads the graphs directly and learns a classification function. Two main challenges associated with this approach are: 1) extracting useful features for classification purpose characterizing the rich information encoded in a graph and 2) reading the graphs sequentially in a significant and logical order. Addressing these challenges is essential, particularly while handling complex structures like CPSs. In a CPS, the relationships between nodes in a graph represent various components, and the edges represent complex

interactions with underlying semantics. A localized graph convolution model has been implemented to address the first challenge, along with its association to graph kernels. The SortPooling layer (Zhang *et al.*, 2018) has been used to address the second challenge, which sorts graph vertices in a logical order so that regular neural networks can be trained on the graphs. The SortPooling layer is essential to sort the vertex features in a coherent order. The sorted data can later be used by traditional convolutional and dense layers to perform additional analysis. In the case of graphs, vertices can be sorted based on their structural roles within the graph in contrast to image and text classification, which has natural ordering. The graph labeling introduced in (Zhang *et al.*, 2018), has been used to sort vertices in a preprocessing step, which is based on the Weisfeiler-Lehman (WL) algorithm (Shervashidze *et al.*, 2011). Experiments on benchmark graph datasets demonstrate that the proposed architecture achieves excellent performance by classifying the graphs based on domain partitioning.

The contributions are as follows: Section 8.1.1 introduces the Graph Nets library that defines the graph structure. Section 8.1.2 discusses the importance of graphs in physical systems. Section 8.1.3 discusses the MPNN model in terms of graph networks. Section 8.1.4 introduces the neural network model based on edge convolutions to partition the graphs into one of the two available classes.

8.1.1. Graph Nets Library. The Graph Nets library (Battaglia *et al.*, 2018) has been used to define the graph structure. This network takes a graph as input and returns a graph as output. The input graph is composed of edge-level (E), node-level (V), and global-level (u) attributes. The output graph contains updated attributes with the same structure as the input graph. Graph networks are part of the broader family of "graph neural networks" (Scarselli *et al.*, 2008). In this model, a graph is defined as G = (u, V, E) in which u represents a global attribute, $V = \{v_i\}_{i=1:N^v}$ represents the set of nodes, where v_i represents a node's attribute, N^v is the cardinality and $E = \{(e_k, r_k, s_k)\}_{k=1:N^e}$ represents the set of edges, where e_k represents the edge's attribute, N^e is the cardinality of edges, r_k

is the receiver node's index, and s_k is the sender node's index. The functions defined in 8.1 are used to perform update and aggregation operation on the nodes, edges and the graph. These can be formulated as follows:

$$\begin{aligned} e_k^{'} &= \phi^e(e_k, v_{r_k}, v_{s_k}, u) & \bar{e}_i^{'} &= \rho^{e\to v}(E_i^{'}) \\ v_i^{'} &= \phi^v(\bar{e}_i^{'}, v_i, u) & \bar{e}^{'} &= \rho^{e\to u}(E^{'}) \\ u^{'} &= \phi^u(\bar{e}^{'}, \bar{v}^{'}, u) & \bar{v}_i^{'} &= \rho^{v\to u}(V^{'}) \end{aligned} \tag{8.1}$$

where $E_i^{'} = \{(e_k^{'}, r_k, s_k)\}_{r_k=i,k=1:N^e}$, $V^{'} = \{v_i^{'}\}_{i=1:N^v}$, and $E^{'} = \cup_i E_i^{'}$. The ϕ^e is used to compute individual edge updates, the ϕ^v is applied across all nodes to compute individual node updates, and the ϕ^u is the global update which is applied once. The ρ functions each take a set as input, and results in a single element which represents the aggregated information. The important property of the ρ functions is that it is invariant to permutations of the inputs, and takes multiple numbers of arguments such as mean, maximum, and element-wise summation.

Algorithm 3 Algorithm to update the node, edge and global attributes (Battaglia *et al.*, 2018)

Input: Graph dataset G, with edges E, vertices V and global attribute u
Output: Updated Tensors for edges, nodes and global attributes l

function GRAPHNETWORK(E, V, u)
  $e_k^{'} \leftarrow e_k$
  for $i \leftarrow 1$ to N^n **do**
    let $E_i^{'} = \{(e_k^{'}, r_k, s_k)\}$
    $\bar{e}_i^{'} \leftarrow \phi^{e\to v}(E_i^{'})$
    $v_i^{'} \leftarrow \phi^v(\bar{e}_i^{'}, v_i, u)$
  let $V^{'} = \{v^{'}\}_{i=1:N^v}$
  let $E^{'} = \{(e_k^{'}, r_k, s_k)\}_{k=1:N^e}$
  $\bar{e}^{'} \leftarrow \phi^{e\to u}(E^{'})$
  $\bar{v}^{'} \leftarrow \phi^{v\to u}(V^{'})$
  $u^{'} \leftarrow \phi^u(\bar{e}^{'}, \bar{v}^{'}, u)$
  return $(E^{'}, V^{'}, u^{'})$

The neural network model takes graph G as an input and the computations progress from the edge, to the node, and finally to the global level. Figure 8.1 shows the message passing architecture used in this work with its update and aggregation functions. Algorithm 3 which is a modified version of the algorithm proposed in (Battaglia *et al.*, 2018) shows the following steps of computation:

- In this algorithm, ϕ^e is not applied to the edges, and e_k is assigned to $e_k^{'}$ for to match the convention of the equations in 8.1. This results in a set of per-edge outputs for each node, i, given by $E_i^{'} = \{(e_k^{'}, r_k, s_k)\}_{r_k=i,k=1:N^e}$ whereas $E^{'} = \cup_i E_i^{'}$ contains the set of all per-edge outputs. For example, in an aircraft example this corresponds to the altitude value that is being passed from one node to another.

- In the next step, $\rho^{e\rightarrow v}$ is applied to $E_i^{'}$, that aggregates the edge updates for all the edges that project to node i, and results in $\bar{e}_i^{'}$. This edge update $\bar{e}_i^{'}$ will be used in the next step's node update. In this way all the edge updates that project onto altimeter are used to update the altimeter node based on the consistency of the edge attribute e_k.

- ϕ_v is applied to each node i, which computes updated node attribute, $v_i^{'}$. This results in a set of per-node outputs, $V^{'} = \{v_i^{'}\}_{i=1:N^v}$. In this step, each node's attribute values are updated to either of the three values 1, -1, and 0, where 1 represents edge attribute values are consistent, -1 represents edge attribute values are inconsistent, whereas 0 represents that the edge attribute does not exist.

- In the next step, $\rho^{e\rightarrow u}$ is used to aggregate all edge updates. This function is applied to $E^{'}$, and the result $\bar{e}^{'}$ will be used to compute the global update.

- $\rho^{v\rightarrow u}$ is used to aggregate all node updates. This function is applied to $V^{'}$, and the result $\bar{v}^{'}$ will be used to compute the global update.

- In the last step, ϕ^u is applied once per graph to compute global attribute update, u'. In the aircraft example, all the values associated with edge labels are aggregated to result in E' and the node updates result in V' which are then used in the global update process, u', to update the class of the graph ($Secure$ or $Non-Secure$).

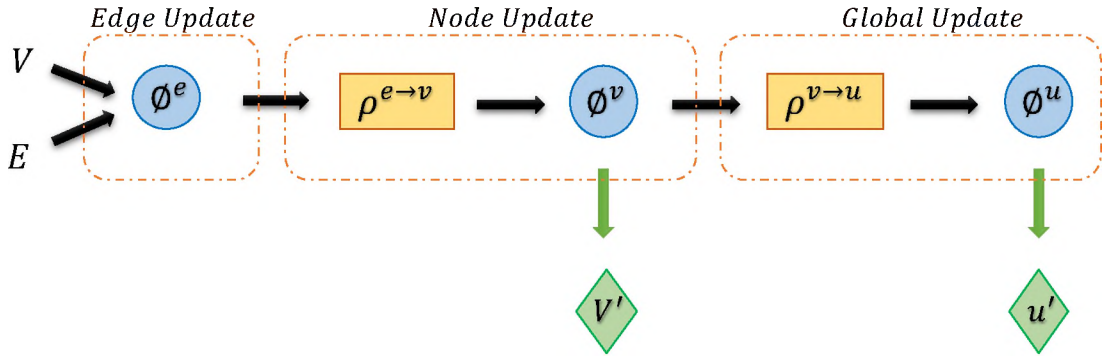


Figure 8.1. Message Passing Architecture

8.1.2. Importance of Graphs. Graphs can predominantly express arbitrary relationships among entities, which implicitly enables the model's input to determine how entities interact and stay isolated. For example, an edge between two nodes represents a relationship, whereas the absence of an edge represents that there is no relationship and cannot influence each other. The entities and relationships in a graph are represented by sets, which are invariant to permutations. The graph networks being invariant to the order of the input elements is often a desirable property. In addition, the node and edge update functions are reused across all nodes and edges, respectively. This means every edge and node in the graph has knowledge about other edges and nodes respectively which helps in computing

the global feature. This is an essential element which served as a motivation to partition the security domains in a CPS by having the knowledge about every other component and interactions between the components.

8.1.3. Message Passing Neural Network (MPNN). This section explains the MPNN algorithm in terms of graph networks. In this section, the input is assumed to explicitly specify the relational structure. Examples of data with such an assumption include physical systems, knowledge graphs, optimization problems, social networks, chemical graphs, parse trees, and road networks with known interactions.

An adjacency matrix$A$ and a set of feature vectors for the nodes are given as inputs to the MPNN model. The adjacency matrix A is comprised of vector-valued entries to indicate the information flow between different nodes in the graph. The edge and node outputs typically are in the form of lists of vectors or tensors, one per edge or node, and the global outputs are in the form of a single vector or tensor. This provides a flexible way for a graph network's output to be passed to other deep learning networks such as CNNs. MPNN proposed in (Gilmer *et al.*, 2017) has similar structure to graph nets architecture discussed in 8.1.1. The only difference is that the readout function used in (Gilmer *et al.*, 2017) doesn't take E' as an input.

8.1.4. Message Passing Edge Convolutional Neural Network (MPENN). This section describes the various components of the proposed MPENN algorith to classify the graphs into different classes (different domains in case of CPS) based on edge, node and global updates. MPENN is comprised of three sequential stages as shown in Figure 8.2:

- The vertices' local substructure features are extracted by the graph convolution layers based on the edge features.
- Node features are computed and updated based on the consistency of the edge features, which are then used to update the global feature.

- The final output predictions are made by the traditional convolutional and dense layers by processing the updated graph representations.

8.1.4.1. Message functions. The following sections illustrate the functions used in the MPENN model to compute node and edge features.

- *Matrix Multiplication:* The message function is defined as $M(h_v, h_w, e_{vw}) = A_{e_{vw}} h_w$ where h_v and h_w represent node features, e_{vw} represent edge features and A represents adjacency matrix.

- *Edge Network:* The authors in (Gilmer *et al.*, 2017) proposed a message function to support vector-valued edge features. This function is represented by $M(h_v, h_w, e_{vw}) = A(e_{vw})h_w$ where $A(e_{vw})$ is a neural network which maps the edge vector e_{vw} to a d-dimensional square matrix.

- *Pair Message:* In general, using matrix multiplication rule, the message from node w to node v is not dependent on the hidden state h_v and is a function only of the edge e_{vw} and the hidden state h_w. The messages should depend on both the source and destination nodes to make an efficient utilization of the network channels in a network. To achieve this, the message function described in (Battaglia *et al.*, 2016) has been used. Therefore, the modified message function is $m_{wv} = f(h_w^t, h_v^t, e_{vw})$ which represents the message from w to v accompanying edge e, where f is a neural network.

8.1.4.2. Graph convolutional layers. The graph convolutional layers (Zhang *et al.*, 2018) takes a graph G and its node information is encoded in the matrix $\mathbf{X} \in \mathbb{R}_{n \times c}$ as inputs, where n is the input and c is the number of features. The graph convolution layer takes the following form:

$$\mathbf{Z}_{t+1} = f(\tilde{D}^{-1}\tilde{G}Z^t W^t) \tag{8.2}$$

where f is a non-linear activation function, $\tilde{G}$ = G + I represents the adjacency matrix of the graph, $\tilde{D}$ represents the diagonal matrix, and $\mathbf{Z} \in \mathbb{R}_{n \times c}$ represents the output activation matrix. The following four steps discusses the graph convolution operation:

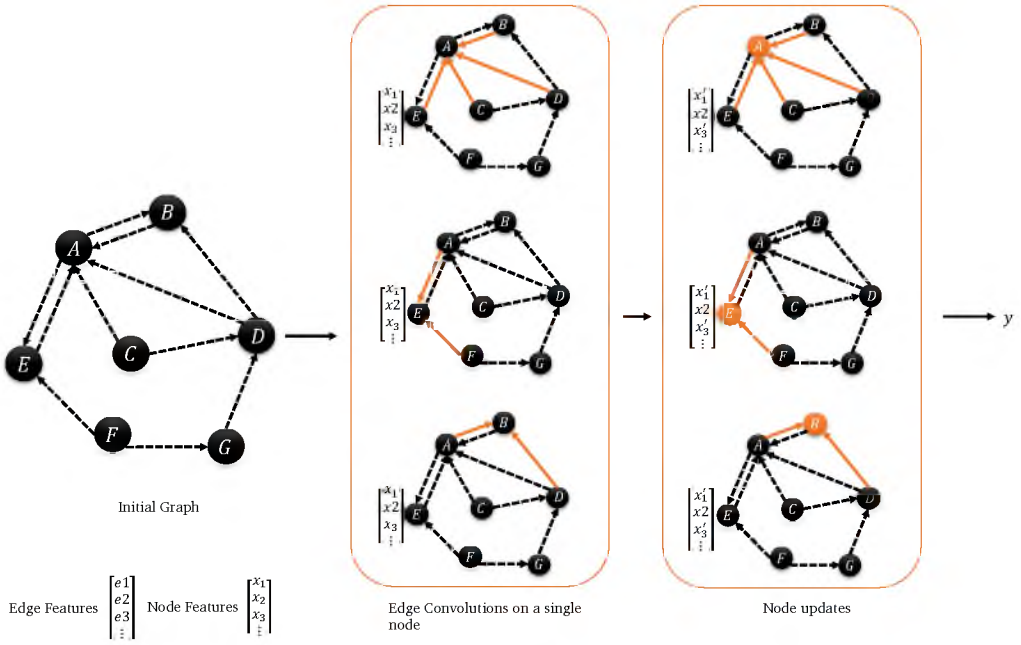


Figure 8.2. MPENN Architecture

- A linear function to transform the features is applied to the node information matrix, denoted by **XW**. This transformation maps the input feature channels c to c' channels in the following layer. The filter weights used in the transformation **W**, are shared among all vertices.

- In the next step, the node information is propagated to neighboring vertices as well as the node itself using $\tilde{A}Y$ where $Y = XW$.

- In the next step, each row i is multiplied by $\tilde{D}_{ii}^{-1}$. This is done to maintain a uniform feature scale after graph convolution.

- In the final step, a pointwise non-linear activation function f is applied which outputs the results of the graph convolutions.

The graph convolution extracts local substructure information by aggregating node information in local neighborhoods. This enables the node information to be shared across all the participating nodes and identify any discrepancy in the data.

8.2. EXPERIMENTAL RESULTS

Three benchmark datasets have been used to evaluate the performance of the MPENN model. All the three datasets have two class labels each representing certain behaviour. To achieve the intended partitioning functionality based on edge and node features, a few instances of the data belonging to a particular class label have been modified to represent anomalies. The code and data are available at https://github.com/anusha_at/MPENN.

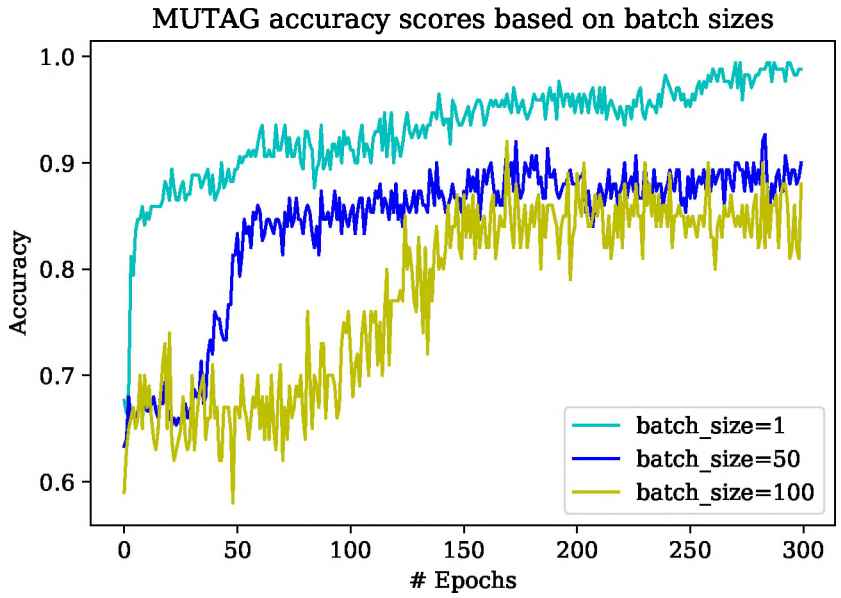


Figure 8.3. MUTAG - Accuracy

8.2.1. Datasets. The benchmark datasets from various fields to compare and analyze the graph classification accuracy of MPENN using graph kernels are NCI1, NCI109, and MUTAG. NCI1 and NCI109 datasets are composed of chemical compounds that are screened to identify non-small cell lung cancer and ovarian cancer cell lines, respectively (Wale *et al.*, 2008). MUTAG (Debnath *et al.*, 1991) is a data set with a collection of 188

mutagenic aromatic and heteroaromatic nitro compounds labelled according to whether they have a mutagenic effect on a specific bacterium called Gramnegtive bacterium *Salmonella typhimurium*. All these datasets are vertex and edge labeled and has two class labels.

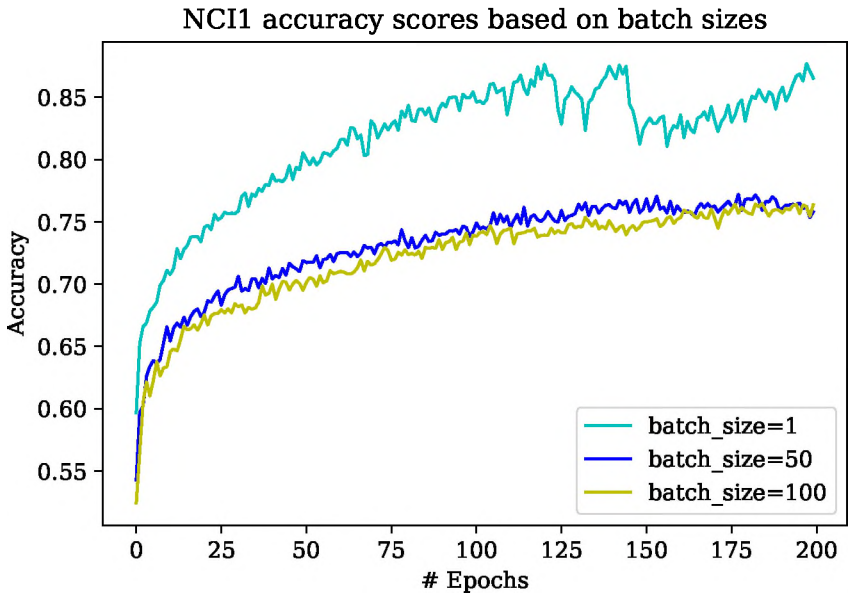


Figure 8.4. NCI1 - Accuracy

8.2.2. Configuration. As described in Section 8.1.4, MPENN-network has four layers. Its configuration is described as *Conv-32→Conv-32→Conv-32→MP→GAP→FC-32→D-0.1→FC-1*, where $Conv - o$ denotes a convolutional layer using batch normalization with o output channels, $D - p$ is a dropout with probability p, $ReLU$ is the activation function with dropout probability of 0.2, MP denotes max-pooling, GAP denotes global average pooling, and $FC - o$ denotes fully connected layer with o output channels. Labels are encoded as one-hot vectors. The MPENN model is trained with stochastic gradient descent (SGD) and cross-entropy loss for 300 epochs with batch sizes 1, 50 and 100 and a variable learning rate starting from 0.001 and optimized after 50, 100, and 150 epochs.

8.2.3. Metrics. This section presents the metrics that are used to measure and compare the performance of the MPENN model.

- *Confusion Matrix:* The Confusion matrix is one of the most intuitive and widely used metrics in deep learning for finding the model's correctness and accuracy. It is used for classification problems in which the output belongs to one of the two or more associated classes.

  - *True Positives (TP):* True positives indicate instances in which the actual class of the data instance and the predicted class is True.
  - *True Negatives (TN):* True negatives indicate instances in which the actual class of the data instance and the predicted class is False.
  - *False Positives (FP):* False positives indicate instances in which the actual class of the data instance is False, and the predicted class is True.
  - *False Negatives (FN):* False refusals indicate instances in which the actual class of the data instance is Dependable, and the predicted class is False.

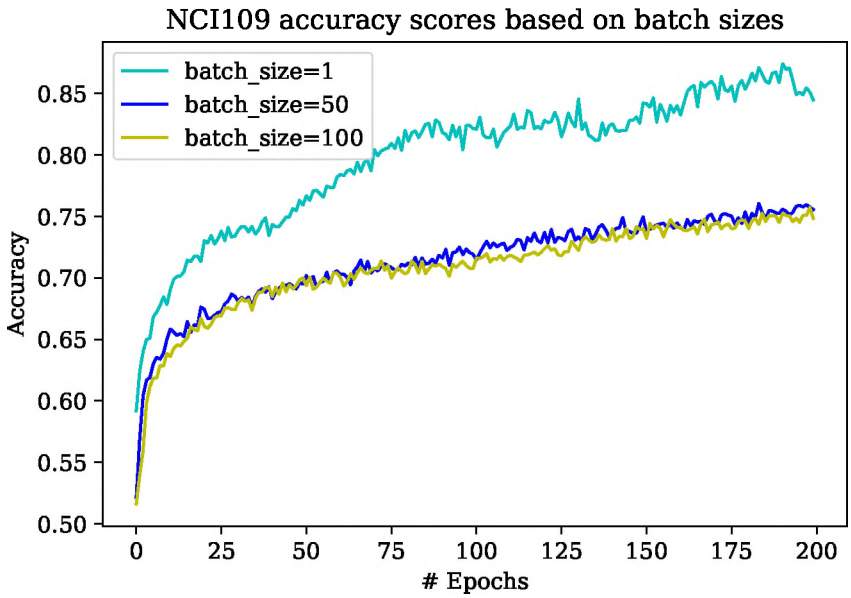


Figure 8.5. NCI109 - Accuracy

- *Accuracy:* Accuracy is used to measure the total number of correct predictions made out of all the predictions made by the model. Accuracy is given by,

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{8.3}$$

- *Loss:* The loss function is an important parameter to calculate the error of the model during the optimization process. Cross-entropy is the most commonly used loss function in deep learning. Cross-entropy is from the information theory field, based on entropy, and usually calculates the difference between two probability distributions. The cross-entropy loss is given by,

$$L(y_i, \hat{y}_i) = -\frac{1}{N}\sum_{i=1}^{N} y_i.log(p(y_i)) + (1 - y_i).log(1 - p(y_i)) \tag{8.4}$$

where y_i represents the actual label, $\hat{y}_i$ represents the predicted label, $p(y)$ is the probability of the label being the actual label, N is the size of the input,

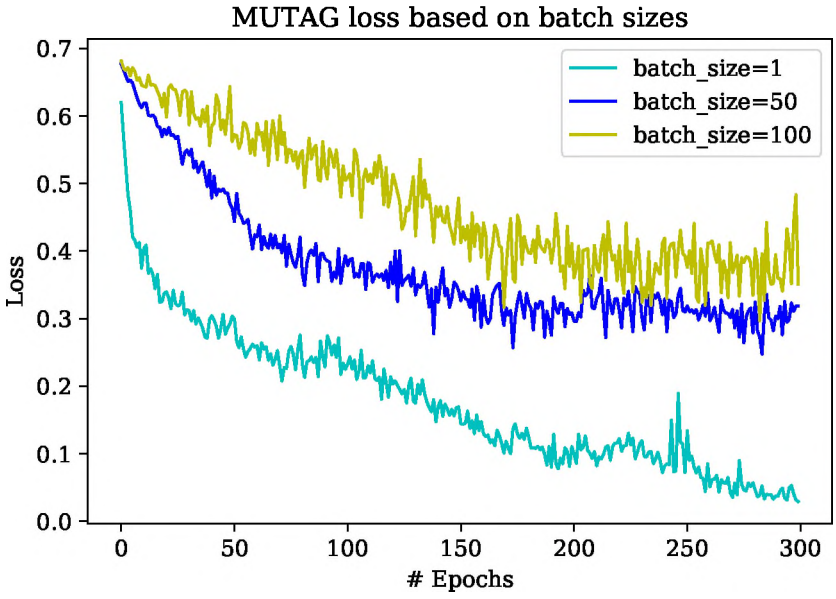


Figure 8.6. MUTAG - Loss

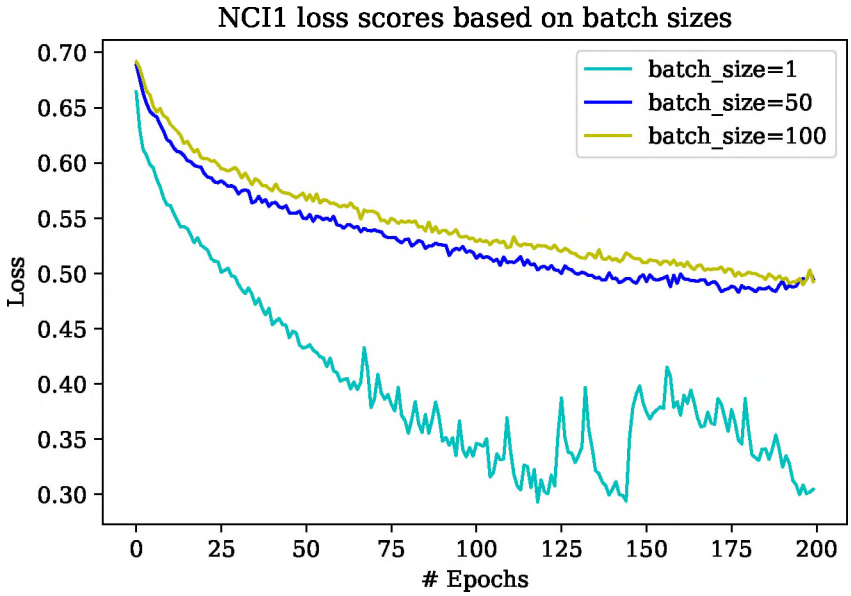


Figure 8.7. NCI1 - Loss

Table 8.1. Results on MUTAG, NCI and NCI109. These datasets were evaluated for accuracy and cross-entropy loss using MPENN.

Dataset	Batch Size	Epoch	Accuracy	Loss
NCI1	1	200	0.86	0.29
NCI109	1	200	0.91	0.21
MUTAG	1	300	0.8	0.24
NCI1	50	200	0.75	0.49
NCI109	50	200	0.5	0.74
MUTAG	50	300	0.9	0.31
NCI1	100	200	0.75	0.49
NCI109	100	200	0.5	0.74
MUTAG	100	300	0.88	0.35

8.2.4. Results. Table 8.1 exhibits that MPENN model performs with an accuracy score of 87.33% for edge-labeled and node-labeled datasets to classify the graphs based on edge and node features. Figures 8.3 - 8.8 demonstrates the performance of the model over 300 epochs in terms of *accuracy* and *loss*. Each dataset performs well at different

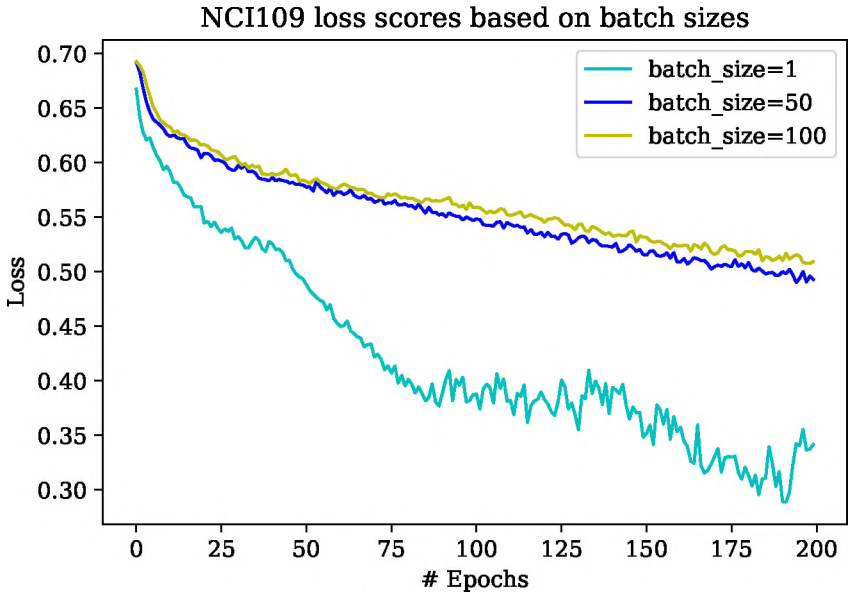


Figure 8.8. NCI109 - Loss

hyperparameter settings, and this can be chosen based on the performance of the model and the dataset. The results demonstrate the importance of exploiting edge labels to identify the anomalies and classify the graphs using convolution-based methods. The model works by exploiting the edge feature to update the node features and then finally updating the global feature to either one of the two class labels associated with the respective dataset. Each dataset is characterized by different attributes, such as protein and chemical compounds. The model achieved good performance by classifying the graphs based on the anomalies injected into the data. For example, the MUTAG dataset is composed of different components, each consisting of multiple subgraphs. These subgraphs are fed into the model to perform edge and node updates. As discussed in Section 8.1.1, these node updates are used to update the global state to either of the two class labels associated with the dataset (aromatic or non-aromatic). In this case, the non-aromatic instances of the dataset are composed of inconsistent edge labels representing a discrepancy representing anomalies in a CPS. The MPENN model performs updates on these subgraphs to classify them based on the node

and global updates. These results show that MPENN can effectively classify the graphs and can be used as a general mechanism to classify the graphs based on anomalies present in the edge features. Two other datasets NCI1 and NCI106 have been used to make sure that this model generalizes well to other datasets as well. Based on these results, this model can be used to identify and classify the security domains in a CPS system with proper hyperparameter settings, identify the security vulnerabilities in a CPS and perform run-time anomaly detection to identify anomalies caused by various factors, as mentioned in Section 4.

9. CONCLUSIONS

This section discusses the broader understandings and assumptions of this dissertation, in addition to the technical specifications in sections 6 to 8. This section is divided into three sections: Section 9.1 presents the scope of applicability of the proposed approach to CPS. Section 9.2 summarizes the limitations of the approach, including the concerns of its practicality. Finally, Section 9.3 presents the directions of future work enabled by this dissertation.

9.1. SUMMARY OF THE PROPOSED APPROACHES

Despite the extensive research in CPS security, there is a need to identify and mitigate the attacks in CPS. In this dissertation, an automated MSDND model is designed and implemented. MSDND is applied to various attack scenarios targeting a specific component failure in an aircraft. MSDND works by identifying independent information flow paths and partitioning them into different security domains based on the consistency of the information flow. The model proposed in this paper is suitable for attacks and failures in a CPS with complex state transitions.

From a conceptual point of view, the graph-based model performs the following functions:

- Detect – Determine the presence of an anomaly or potential failures in the information flow paths.
- Evaluate – Assess the impact of the anomaly based on the valuation functions.
- Prioritize – Determine if removing the affected information flow path affects the system.

- Verify – Check if mitigation is possible by examining the nodes that have valuation functions.
- Action – Partition the information flow paths into Secure and Non-Secure domains.

9.2. LIMITATIONS

Implementing a generalized security mechanism for a wide variety of CPS is a difficult task. Our approach aims at solving this problem by implementing a graph-based model and the features associated with the edges and vertices are hand-crafted. More research can be done in this area to come up with automatic feature recognition and link predictions.

9.3. FUTURE WORK

In moving forward, the focus would be on the following:

- Extend the MSDND analysis to other infrastructures and monitor if all the attack scenarios are identified.
- Use MSDND to model confidentiality, integrity and availability vulnerabilities.
- Make MSDND analysis scalable to extend this to all the critical infrastructures to identify and evaluate cyber-physical risks.
- Apply graph neural networks to automatically construct the graph and identify the anomalies. This is done by clustering the components in the system based on the validation of the information flow paths (the information flow paths qualified as secure will be in secure security domain and the other information flow paths will be in non-secure security domain).

- Use graph neural networks to construct the graphs from architecture diagrams of a CPS and perform link prediction to connect different components.
- Use a feature extraction model to label the nodes and vertices in a graph based on neighbor interactions and their features.

REFERENCES

"Characterization of a cusum model-based sensor attack detector, author=Murguia, Carlos and Ruths, Justin," in "2016 IEEE 55th Conference on Decision and Control (CDC)," IEEE, 2016 pp. 1303–1309.

Administration, F. A., "Pilot's Handbook of Aeronautical Knowledge," 2016, [Online; posted 2016].

Administration, F. A., "Advanced Avionics Handbook," 2017, [Online; posted 2017].

Amin, S., Cárdenas, A. A., and Sastry, S. S., "Safe and Secure Networked Control Systems Under Denial-of-Service Attacks," in "International Workshop on Hybrid Systems: Computation and Control," Springer, 2009 pp. 31–45.

Battaglia, P., Pascanu, R., Lai, M., Rezende, D. J., *et al.*, "Interaction networks for learning about objects, relations and physics," in "Advances in neural information processing systems," 2016 pp. 4502–4510.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., *et al.*, "Relational inductive biases, deep learning, and graph networks," arXiv preprint arXiv:1806.01261, 2018.

Behrmann, G., David, A., and Larsen, K. G., "A tutorial on UPPAAL," in "Formal methods for the design of real-time systems," Springer, 2004 pp. 200–236.

Blackburn, P. and Van Benthem, J., "Modal logic: a semantic perspective," in "Studies in Logic and Practical Reasoning," volume 3, pp. 1–84, Elsevier, 2007.

Blackburn, P., van Benthem, J. F., and Wolter, F., *Handbook of modal logic*, Elsevier, 2006.

Bohnenkamp, H. C., d'Argenio, P. R., Hermanns, H., and Katoen, J. P., "Modest: A compositional modeling formalism for real-time and stochastic systems," 2004.

Cárdenas, A. A., Amin, S., Lin, Z.-S., Huang, Y.-L., Huang, C.-Y., and Sastry, S., "Attacks against process control systems: risk assessment, detection, and response," in "Proceedings of the 6th ACM symposium on information, computer and communications security," 2011 pp. 355–366.

Cárdenas, A. A., Amin, S., and Sastry, S., "Research Challenges for the Security of Control Systems," in "HotSec," 2008a .

Cárdenas, A. A., Amin, S., and Sastry, S., "Secure control: Towards survivable cyber-physical systems," in "2008 The 28th International Conference on Distributed Computing Systems Workshops," IEEE, 2008b pp. 495–500.

Cho, H. S. and Woo, T. H., "Cyber security in nuclear industry–Analytic study from the terror incident in nuclear power plants (NPPs)," Annals of Nuclear Energy, 2017, **99**, pp. 47–53.

Clarke, E. M. and Wing, J. M., "Formal methods: State of the art and future directions," ACM Computing Surveys (CSUR), 1996, **28**(4), pp. 626–643.

Conrad, E., Misenar, S., and Feldman, J., *CISSP study guide*, Newnes, 2012.

de Oliveira, A. B., Netz, P. A., Colla, T., and Barbosa, M. C., "Structural anomalies for a three dimensional isotropic core-softened potential," The Journal of chemical physics, 2006, **125**(12), p. 124503.

Debnath, A. K., Lopez de Compadre, R. L., Debnath, G., Shusterman, A. J., and Hansch, C., "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity," Journal of medicinal chemistry, 1991, **34**(2), pp. 786–797.

Dibaji, S. M., Pirani, M., Flamholz, D. B., Annaswamy, A. M., Johansson, K. H., and Chakrabortty, A., "A systems and control perspective of CPS security," Annual Reviews in Control, 2019.

Elkaduwe, D., Klein, G., and Elphinstone, K., "Verified protection model of the seL4 microkernel," in "Working Conference on Verified Software: Theories, Tools, and Experiments," Springer, 2008 pp. 99–114.

Fisher, K., "Using formal methods to enable more secure vehicles: DARPA's HACMS program," in "Proceedings of the 19th ACM SIGPLAN international conference on Functional programming," 2014 pp. 1–1.

Gallagher, R., "A guide to understanding security modeling in trusted systems," National Computer Security Center, USA, NCSC-TG-010, 1992.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E., "Neural message passing for quantum chemistry," in "Proceedings of the 34th International Conference on Machine Learning-Volume 70," JMLR. org, 2017 pp. 1263–1272.

Goguen, J. A. and Meseguer, J., "Security policies and security models," in "1982 IEEE Symposium on Security and Privacy," IEEE, 1982 pp. 11–11.

Goldblatt, R., "Mathematical modal logic: A view of its evolution," in "Handbook of the History of Logic," volume 7, pp. 1–98, Elsevier, 2006.

Greer, S. C. and Moldover, M. R., "Thermodynamic anomalies at critical points of fluids," Annual Review of Physical Chemistry, 1981, **32**(1), pp. 233–265.

Hawkins, D. M., *Identification of outliers*, volume 11, Springer, 1980.

Howser, G. and McMillin, B., "A Multiple Security Domain Model of a Drive-by-Wire System," in "Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual," IEEE, 2013 pp. 369–374.

Howser, G. and McMillin, B. M., "Using Information-Flow Methods to Analyze the Security of Cyber-Physical Systems," IEEE Computer, 2017, **50**(4), pp. 17–26, doi: 10.1109/MC.2017.112.

Huitsing, P., Chandia, R., Papa, M., and Shenoi, S., "Attack taxonomies for the Modbus protocols," International Journal of Critical Infrastructure Protection, 2008, **1**, pp. 37–44.

Humayed, A., Lin, J., Li, F., and Luo, B., "Cyber-Physical Systems Security—A Survey," IEEE Internet of Things Journal, 2017, **4**(6), pp. 1802–1831.

Katoen, J.-P., Zapreev, I. S., Hahn, E. M., Hermanns, H., and Jansen, D. N., "The ins and outs of the probabilistic model checker MRMC," Performance evaluation, 2011, **68**(2), pp. 90–104.

Kim, Y., Jo, J.-Y., and Lee, S., "ADS-B Vulnerabilities and a Security Solution with a Timestamp," IEEE Aerospace and Electronic Systems Magazine, 2017, **32**(11), pp. 52–61.

Kravchik, M. and Shabtai, A., "Detecting cyber attacks in industrial control systems using convolutional neural networks," in "Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy," 2018 pp. 72–83.

Lamnabhi-Lagarrigue, F., Annaswamy, A., Engell, S., Isaksson, A., Khargonekar, P., Murray, R. M., Nijmeijer, H., Samad, T., Tilbury, D., and Van den Hof, P., "Systems & Control for the future of humanity, research agenda: Current and future roles, impact and grand challenges," Annual Reviews in Control, 2017, **43**, pp. 1–64.

Leroy, X. *et al.*, "The CompCert verified compiler," Documentation and user's manual. INRIA Paris-Rocquencourt, 2012, **53**.

Liu, Y., Ning, P., and Reiter, M. K., "False Data Injection Attacks Against State Estimation in Electric Power Grids," ACM Transactions on Information and System Security (TISSEC), 2011, **14**(1), p. 13.

Lowe, G., "Semantic models for information flow," Theoretical Computer Science, 2004, **315**(1).

Manesh, M. R. and Kaabouch, N., "Analysis of Vulnerabilities, Attacks, Countermeasures and Overall Risk of the Automatic Dependent Surveillance-Broadcast (ADS-B) System," International Journal of Critical Infrastructure Protection, 2017, **19**, pp. 16–31.

McCallie, D., Butts, J., and Mills, R., "Security Analysis of the ADS-B Implementation in the Next Generation Air Transportation System," International Journal of Critical Infrastructure Protection, 2011, **4**(2), pp. 78–87.

McCumber, J., "Information systems security: A comprehensive model," in "Proceedings 14th National Computer Security Conference," 1991 pp. 328–337.

McLaughlin, S., Konstantinou, C., Wang, X., Davi, L., Sadeghi, A.-R., Maniatakos, M., and Karri, R., "The cybersecurity landscape in industrial control systems," Proceedings of the IEEE, 2016, **104**(5), pp. 1039–1057.

McLean, J., "Security models and information flow," Technical report, Naval Research Lab, Washington DC, Center For High Assurance Computing Systems, 1990.

Miller, B. and Rowe, D., "A survey SCADA of and critical infrastructure incidents," in "Proceedings of the 1st Annual conference on Research in information technology," 2012 pp. 51–56.

Mitchell, R. and Chen, I.-R., "A survey of intrusion detection techniques for cyber-physical systems," ACM Computing Surveys (CSUR), 2014, **46**(4), pp. 1–29.

Mo, Y. and Sinopoli, B., "Secure Control Against Replay Attacks," in "2009 47th annual Allerton conference on communication, control, and computing (Allerton)," IEEE, 2009 pp. 911–918.

Myers, A. C., Zheng, L., Zdancewic, S., Chong, S., and Nystrom, N., "Jif: Java information flow," Software release. Located at http://www. cs. cornell. edu/jif, 2001, **2005**.

Pasqualetti, F., Dörfler, F., and Bullo, F., "Cyber-Physical Security via Geometric Control: Distributed Monitoring and Malicious Attacks," in "2012 IEEE 51st IEEE Conference on Decision and Control (CDC)," IEEE, 2012 pp. 3418–3425.

Peterson, J. L., "Petri nets," ACM Computing Surveys (CSUR), 1977, **9**(3), pp. 223–252.

Pottier, F. and Simonet, V., "Information flow inference for ML," in "Proceedings of the 29th ACM SIGPLAN-SIGACT symposium on Principles of programming languages," 2002 pp. 319–330.

Sabelfeld, A. and Myers, A. C., "Language-based information-flow security," IEEE Journal on selected areas in communications, 2003, **21**(1), pp. 5–19.

Sampigethaya, K. and Poovendran, R., "Aviation cyber–physical systems: Foundations for future aircraft and air transport," Proceedings of the IEEE, 2013, **101**(8), pp. 1834–1855.

Sandberg, H., Teixeira, A., and Johansson, K. H., "On security indices for state estimators in power networks," in "First Workshop on Secure Control Systems (SCS), Stockholm, 2010," 2010 .

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G., "The graph neural network model," IEEE Transactions on Neural Networks, 2008, **20**(1), pp. 61–80.

Sen, K., Viswanathan, M., and Agha, G., "Vesta: A statistical model-checker and analyzer for probabilistic systems," in "Second International Conference on the Quantitative Evaluation of Systems (QEST'05)," IEEE, 2005 pp. 251–252.

Shervashidze, N., Schweitzer, P., Van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M., "Weisfeiler-lehman graph kernels," Journal of Machine Learning Research, 2011, **12**(77), pp. 2539–2561.

Simonet, V. and Rocquencourt, I., "Flow Caml in a nutshell," in "Proceedings of the first APPSEM-II workshop," 2003 pp. 152–165.

Simonovsky, M. and Komodakis, N., "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in "Proceedings of the IEEE conference on computer vision and pattern recognition," 2017 pp. 3693–3702.

Strohmeier, M., Schafer, M., Lenders, V., and Martinovic, I., "Realities and Challenges of NextGen Air Traffic Management: The Case of ADS-B," IEEE Communications Magazine, 2014, **52**(5), pp. 111–118.

Stroud, R., "On the relationship between fault tolerance and intrusion tolerance: experience from the MAFTIA project," 2003.

Sutherland, D., "A model of information," in "Proceedings of the 9th national computer security conference," volume 247, Washington, DC, 1986 pp. 175–183.

Teixeira, A., Amin, S., Sandberg, H., Johansson, K. H., and Sastry, S. S., "Cyber Security Analysis of State Estimators in Electric Power Systems," in "49th IEEE conference on decision and control (CDC)," IEEE, 2010 pp. 5991–5998.

Teixeira, A., Shames, I., Sandberg, H., and Johansson, K. H., "A secure control framework for resource-limited adversaries," Automatica, 2015, **51**, pp. 135–148.

Thudimilla, A. and McMillin, B., "Multiple Security Domain NonDeducibility Air Traffic Surveillance Systems," in "2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)," IEEE, 2017 pp. 136–139.

Wale, N., Watson, I. A., and Karypis, G., "Comparison of descriptor spaces for chemical compound retrieval and classification," Knowledge and Information Systems, 2008, **14**(3), pp. 347–375.

Wang, D., Wang, Z., Shen, B., Alsaadi, F. E., and Hayat, T., "Recent advances on filtering and control for cyber-physical systems under security and resource constraints," Journal of the Franklin Institute, 2016, **353**(11), pp. 2451–2466.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y., "A comprehensive survey on graph neural networks," IEEE Transactions on Neural Networks and Learning Systems, 2020.

Yang, H., Yao, M., Xu, Z., and Liu, B., "LHCSAS: A Lightweight and Highly-Compatible Solution for ADS-B Security," in "GLOBECOM 2017-2017 IEEE Global Communications Conference," IEEE, 2017 pp. 1–7.

Yang, H., Zhou, Q., Yao, M., Lu, R., Li, H., and Zhang, X., "A Practical and Compatible Cryptographic Solution to ADS-B Security," IEEE Internet of Things Journal, 2018, **6**(2), pp. 3322–3334.

Younes, H. L., "Verification and planning for stochastic processes with asynchronous events," Technical report, Carnegie-Mellon Univ Pittsburgh PA, School of Computer Science, 2005.

Zdancewic, S., "Challenges for information-flow security," in "Proceedings of the 1st International Workshop on the Programming Language Interference and Dependence (PLID'04)," 2004 p. 6.

Zhang, M., Cui, Z., Neumann, M., and Chen, Y., "An end-to-end deep learning architecture for graph classification," in "Thirty-Second AAAI Conference on Artificial Intelligence," 2018 .

VITA

Anusha Thudimilla was born in Hyderabad, India. She received her B.Tech. in Information Technology from the Mahaveer Institute of Science and Technology in May 2012. She graduated with a Master of Science in Computer Science from Missouri University of Science and Technology in December, 2017. She received a Doctor of Philosophy in Computer Science from Missouri University of Science and Technology in August, 2020.

After she moved to Missouri, Anusha worked as a student employee for the Miner Phonathon at Missouri S&T. While there she greatly enjoyed her work as a research assistant under Dr. Bruce McMillin for two years. In addition to publishing the papers presented in this dissertation, Anusha presented her research at scientific meetings and participated in conferences discussing challenges in high performance computing. This was possible as a result of her securing competitive funding from the National Science Foundation. Anusha earned multiple competition prizes for presentations given at the Missouri S&T Computer Science department and the High-Assurance Systems Engineering (HASE) conference.

After MS graduation, she worked for Ungerboeck International as a Software Engineer. She later worked at ReachMobi Inc. as a full-stack developer while pursuing the Ph.D. as a part-time student. In 2018, she left the industry to become a full-time Ph.D. student. She interned as a Research Engineer at American Express during her Ph.D.