Fall 2017

Multiple security domain nondeducibility air traffic surveillance systems

Anusha Thudimilla

MULTIPLE SECURITY DOMAIN NONDEDUCIBILITY
AIR TRAFFIC SURVEILLANCE SYSTEMS

by

ANUSHA THUDIMILLA

A THESIS

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

in

COMPUTER SCIENCE

2017

Approved by

Dr. Bruce McMillin, Advisor
Dr. Daniel Tauritz
Dr. Jennifer Leopold

ABSTRACT

Traditional security models partition the security universe into two distinct and completely separate worlds: high and low level. However, this partition is absolute and complete. The partition of security domains into high and low is too simplistic for more complex cyber-physical systems (CPS). Absolute divisions are conceptually clean, but they do not reflect the real world. Security partitions often overlap, frequently provide for the high level to have complete access to the low level, and are more complex than an impervious wall. The traditional models that handle situations where the security domains are complex or the threat space is ill defined are limited to mutually exclusive worlds. These models are limited to accepting commands from a single source in a system but the CPS accepts commands from multiple sources.

This paper utilizes Multiple Security Domain Nondeducibility (MSDND) as a model to determine information flow among multiple partitions, such as those that occur in a CPS. MSDND is applied to selected aspects of Traffic Collision and Avoidance System (TCAS) and Automatic Dependent Surveillance-Broadcast (ADS-B) air traffic surveillance systems under various physical and cyber security vulnerabilities to determine when the actual operational state can, and cannot be, deduced. It is also used to determine what additional information inputs and flight physics are needed to determine the actual operational state. Several failure scenarios violating the integrity of the system are considered with mitigation using invariants.

ACKNOWLEDGMENTS

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

Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Daniel Tauritz and Dr. Jennifer Leopold, 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 Prakash Rao Dunaka and Uday Ganesh Kanteti for their inputs, feedback, and cooperation. Without their passionate participation and input, this project 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 Dawn Davis and Rhonda Grayson 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 thesis. This accomplishment would not have been possible without them. Thank you.

A very special gratitude goes out to all down at National Institute of Standards and Technology, Grant Number 60NANB15D236 and with support from the Missouri S&T Intelligent Systems Center and the US National Science Foundation, Award Number CNS-1505610 for helping and providing the funding for the work.

TABLE OF CONTENTS

Page

SECTION

APPENDICES

LIST OF ILLUSTRATIONS

LIST OF TABLES

Table Page

NOMENCLATURE

SYMBOL	DESCRIPTION
s_x	A boolean state variable, x is true or false
W	The set of all possible worlds of the system
w	A world of interest
R	Transition
F	Frame
M	Model
SD	Security Domain
df	Tests whether event variable has a defined value
ϕ	A boolean statement that can be evaluated
V_x^i	A valuation function of boolean x in domain i
p, p_1, p_2	Pilot
c	Controller
t, t_1, t_2	TCAS reading
r	ATC radar
x	TCAS transponder
v	INS velocity
d	INS distance
a	Altimeter
ps	Pitot static system
L	Airspeed value of LRI
δ	Altitude of alternate static source
a_1, a_2	ADS-B-1 and ADS-B-2
i	Invariant source

1. INTRODUCTION

The aviation industry is going through a major transformation to meet increasing air traffic, societal, and business demands. Major transformation examples include the e-enabled aircraft and next-generation air transport system [2, 3].

The e-enabled aircraft is envisioned as an intelligent node with seamless mobility in a global network of ground, air, space infrastructures [4, 5, 6, 7]. The Boeing B787 is a seminal example which combines the power of integrated information and communications systems to drive operational efficiency, enhance revenue, and streamline airplane maintenance. Next-generation air transport systems are aimed at applying the cyber aspect to infrastructures, hardware, personnel, and processes.

The increasing safety and security concerns, such as terrorism, deteriorating weather, wildlife collisions, pilot , system malfunction and ground crew fatigue have become a major concern for the Federal Aviation Administration (FAA). CPS [8] such as industrial control systems are examples of such integration where the effects on the physical world are controlled through the use of smart technologies operated by computers [9]. Air transportation systems and airplanes are major CPS instantiations, and safety and security are central requirements for enabling high confidence CPS [10].

Recent FAA regulations indicate that tight cyber-physical integrations, within aircraft and between aircraft and offboard systems, warrant a surgical consideration of cyber-physical interactions and potential performance risks from cyber and physical threats [4]. Hence, this paper proposes information flow security analysis of CPS for a foundational understanding of information flow paths, risks and performance of air traffic surveillance system's cyber-physical integrations.

1.1. CYBER-PHYSICAL SYSTEMS

Cyber-Physical Systems (CPS) is an emerging vision for next-generation information systems that boldly transform the way modern society perceives the physical world, lives, moves, interacts in it, and systems on which human safety and public well being rests. CPS represent a new generation of systems that integrate computing and communication capabilities with the dynamics of physical and engineered systems.

CPS are susceptible to vulnerabilities such as defects in the platform, misconfiguration of the system, improper network connections and malware. Attackers can take advantage of vulnerabilities in CPS to take control of the system. With physical manifestations in the real world, attacks on CPS can cause disruption to physical services or create a national disaster. As a CPS requires a tight coupling between the physical and cyber controlling components, it is crucial to ensure that the system is secure for all the cyber and physical processes. Therefore, protecting the CPS against cyber attacks is of paramount importance [8].

Traditional security methods can be applied to protect a CPS, such as a critical infrastructure system, against cyber threats or threats imposed by malicious insiders and attackers. However, due to the unique characteristics and complexity of a CPS, traditional security models and approaches are insufficient to address the security challenges of a CPS [11, 12, 13, 14, 15, 16]. For example, installing security patches or numerous system updates that require taking the system offline is difficult, not economically justifiable, and often infeasible. Also, new updates or security patches may create other problems such as in a case where a nuclear power plant accidentally was shutdown after a software update [17].

Information flow security in a CPS can lead to particularly complex security partitions i.e more than just high and low. Tools that work well with securing the cyber part of the system rarely work well to keep the physically observable parts of the system from leaking information. Physically locking the fence around the physical parts of the CPS does not protect from a purely cyber attack. Typical electronic or cryptographic solutions do not match specific cases closely enough to handle the cyber-physical interfaces as they are mainly focused on cyber threat prevention and protection. A persistent attacker with enough time and backing will get in.

A recent, comprehensive survey [18] includes 147 references to publications related to information-flow security. The bulk of these papers are concerned with defining and refining variations on noninterference, the fundamental information-flow property that essentially requires that secret information not affect publicly observable behavior of a system. Many of the remaining papers describe approaches to enforcing information-flow policies using program analysis techniques. Yet despite this large body of literature and considerable, ongoing attention from the research community, information-flow based enforcement mechanisms have not been widely used.

The real challenge in information-flow security is not in giving better, more precise definitions of noninterference and related properties for more complicated combinations of language features and system models. Nor is the real challenge implementing languages that support information-flow policies; the programming languages Jif [19], and Flow Caml [20, 21], provide high-level, realistic programming languages with support for sophisticated information-flow controls. Although there are certainly interesting open questions in both its theory and implementation, the real challenge for information-flow security is demonstrating that all of this theory and these language designs are actually useful - the technology needs to be applied to real problems, or, failing that, understand why such an appealing technology is not useful in practice.

This thesis examines the current security models in practice for information-flow technology and tries to identify some of the main obstacles of putting it into practice. It also introduces a new information flow security model to minimize the shortcomings of the traditional models. The developed model is used to analyze various components of avionic systems and present solutions to mitigate the potential problems.

1.2. NONDEDUCIBILITY AND SECURITY MODELS

Nondeducibility (ND) was introduced by Sutherland [22] as an attempt to use modal techniques to model data in a partitioned security system. The possible worlds (e.g., state collections) of this model are partitioned into two disjoint sets and information is restricted to one side of the partition or the other [23]. Information that could not be inferred from the other side of the partition was determined to be Nondeducibility secure. Overlapping security domains break Sutherland's Nondeducibility as do information flows, the correctness of the system cannot be evaluated because of the partitions [24].

A modal technique to model complex security domains, Multiple Security Domain Model Nondeducibility (MSDND) was introduced. MSDND can model any system where Sutherland Nondeducibility holds and complex systems where Nondeducibility cannot be determined. MSDND models CPS well, even when the security domains overlap or the boundaries are not ideal and leak [24]. Modal logic based models work well for systems having valuation functions for the states but the complex cyber-physical systems leak information because the interactions between physical parts of the system can be watched for changes. By their very nature, CPS are messy from a security domain view point. Domains overlap, the boundaries are not clean (ideal boundaries cannot leak information), and outside threats can leak into domains thought to be secure [24].

Computer security tools work best when secure domains are cleanly nested inside less secure domains like a medieval castle with its outer walls and interior keep. This model serves us well for most uses, but breaks down when applied to CPS. Because CPS typically need to secure both data and information flow, the security domain picture gets complicated. There is a need for tools that can model the cyber and physical components of CPS.

1.3. PROVERIF

In recent years, research has strongly shifted from manual proofs to automated proofs of security. The verification step to ensure that a computer program, a protocol or a CPS has certain requested properties is a crucial one, and this task should ideally be done by formal reasoning, rather than by tests and simulations. There are two possible approaches to protocol verification: the formal model and the computational model. The first model is in a highly idealized setting and it can be effectively implemented using fully-automated protocol verifiers. The second approach borrows ideas from complexity theory and requires much more human intervention in proofs, and it is only recently being automated [25]. These verification techniques allow us to uncover design faults that may remain hidden for years. In this thesis, Bruno Blanchet's ProVerif [26, 27, 28] automates the MSDND process and verifies the correctness of the system with the help of proofs.

ProVerif is a tool for automatically analyzing the security of cryptographic protocols. Support is provided for, but not limited to, cryptographic primitives including: symmetric and asymmetric encryption; digital signatures; hash functions; bit-commitment; and non-interactive zero-knowledge proofs. ProVerif is capable of proving reachability properties, correspondence assertions, and observational equivalence. These capabilities are particularly useful to the computer security domain since they permit the analysis of secrecy and authentication properties. Moreover, emerging properties such as privacy, traceability, and verifiability can also be considered. Protocol analysis is considered with respect to an unbounded number of sessions and an unbounded message space. The tool is also capable of attack reconstruction: when a property cannot be proved, ProVerif tries to reconstruct an execution trace that falsifies the desired property.

The primary goal of ProVerif is the verification of cryptographic protocols. Cryptographic protocols are concurrent programs which interact using public communication channels such as the Internet to achieve some security-related objective. These channels are assumed to be controlled by a very powerful environment which captures an attacker with

"Dolev-Yao" capabilities. A second class of models is used by the community of formal methods, and includes typically the Dolev-Yao model [29] and the Spi-calculus [30]. By focusing on the protocol layer, these models aim to account for a variety of attacks resulting from complex interactions between an active attacker and a possibly unbounded number of parallel sessions. Since the attacker has complete control of the communication channels, the attacker may: read, modify, delete, and inject messages. The attacker is also able to manipulate data, for example: compute the *i*th element of a tuple; and decrypt messages if it has the necessary keys. The environment also captures the behavior of dishonest participants; it follows that only honest participants need to be modeled [31]. ProVerif's input language allows such cryptographic protocols and associated security objectives to be encoded in a formal manner, allowing ProVerif to automatically verify claimed security properties. Cryptography is assumed to be perfect; that is, the attacker is only able to perform cryptographic operations when in possession of the required keys. In other words, it cannot apply any polynomial-time algorithm, but is restricted to apply only the cryptographic primitives specified by the user. The relationships between cryptographic primitives are captured using rewrite rules and/or an equational theory.

In this thesis, the deducibility of information flow is verified both manually and using a tool with respect to individual entities to show evidence of possible attacks. This thesis uses a combination of statecharts and modal logic to model complex CPS systems which accept commands from multiple sources. Both statecharts and MSDND are used to analyse the information flow paths in a Traffic Collision and Avoidance System (TCAS) and a Automatic Dependent Broadcast - Surveillance (ADS-B) System. This model is extended to identify the problems and provide feasible solutions in reference to various other avionic systems. It also provides a practical understanding of how cyber security impacts airplane functions, in the presence of existing safety, development, and training requirements and processes. In addition to this, ProVerif is used to automate the MSDND process for the air traffic surveillance systems using observational equivalence and integrity properties.

The remainder of this thesis is organized as follows. Section 2 provides a brief introduction about the attacker model and the various avionics systems used to model MSDND. Section 3 motivates the need for cyber physical security of air traffic surveillance systems. Section 4 presents a brief survey about the contributions made to devise security proofs for ensuring safety and security for the avionic systems and the associated challenges. Section 5 proposes an approach for modeling the air traffic surveillance systems using modal logic. Section 6 provides MSDND proofs for various attack scenarios and the ProVerif code associated with each scenario is present in the Appendices. Section 7 provides possible future work. Section 8 presents conclusions.

2. SYSTEM MODEL

The world of air traffic control (ATC) is moving from uncooperative and independent (primary surveillance radar, PSR) to cooperative and dependent air traffic surveillance (secondary surveillance radar, SSR). This paradigm shift holds the promise of reducing the total cost of deployment and improving the detection accuracy of aircraft.

2.1. ADVERSARY AND ATTACK MODEL

The adversary in this model can send fake data by taking control of the aircraft, cause a malfunction in one of the aircraft components, include manual errors by the pilot and include environmental factors while being unnoticed rather than to disable and disrupt the entire aircraft. Therefore, reconnaissance to know about the system's operation becomes important as attack attempts are considered. The failure in implementing the attack can more easily be detected due to the deterministic and predictable nature of the system.

This thesis assumes that the adversary has an understanding about the air traffic surveillance systems and has knowledge of the system functions. For example, an attacker may obtain control over the aircraft cabin by hijacking the plane and compromising one of the systems. These assumptions are not unreasonable as demonstrated by Stuxnet [32], a multi-stage attack in which the attackers compromised many other systems before reaching their target system, and in which the attackers stayed undetected for months conducting reconnaissance on the target system before they launched their attacks.

2.2. TRAFFIC COLLISION AND AVOIDANCE SYSTEM (TCAS)

TCAS was designed to operate in traffic densities of up to 0.3 aircraft per square nautical mile (nmi); i.e., 24 aircraft within a 5 nmi radius, which was the highest traffic density envisioned over the next 20 years. The main functions of TCAS are to identify a potential collision threat, communicate the detected threat to the pilot, and assist in the resolution of the threat by recommending an avoidance maneuver. This is applied if an air traffic controller (ATC) fails to maintain separation via clearances. The TCAS is a beacon-based airborne collision avoidance system that is able to operate in all airspace without reliance on ground equipment. Figure 2.1 presents how the TCAS interacts with in-flight and ground equipment.

A TCAS installation can conceptually be divided into two subsystems: surveillance and control logic. TCAS works by one aircraft interrogating other aircraft transponders. This way, each TCAS equipped aircraft can locate nearby transponder equipped aircraft, and potential collisions can be detected. Surveillance of the air traffic environment is based on air-to-air interrogations broadcast once per second from antennae on the TCAS aircraft using the same frequency (1030 MHz). Transponders on nearby intruder aircraft receive these interrogations and send replies at 1090 MHz. Two types of transponders are currently in use: Mode-C transponders, which do not have unique addressing capability, and Mode-S transponders, which have a unique 24 bit identifier. To track Mode-C intruders, TCAS transmits "Mode-A, C-only all call" interrogations once per second. All Mode-A, C equipped aircraft in a region around the TCAS aircraft reply. TCAS sends interrogators using a four-beam directional antenna with 90 degree beams. In contrast, Mode-S equipped intruders are tracked with a selective interrogation once per second directed at that specific intruder by listening the squitter. Note that Mode-S transponders send out spontaneous signals known as 56-bit squitters. All aircraft with TCAS are equipped with Mode-S transponders. The TCAS concept makes use of the radar beacon transponders installed on aircraft to operate with ATC's ground-based radars. The level of protection provided by

TCAS equipment depends on the type of transponder the target aircraft is carrying. It should be noted that TCAS provides no protection against aircraft that do not have an operating transponder. Figure 2.2 presents the communication between two aircraft using TCAS.

Without reliance on ground equipment, TCAS is capable of providing resolution advisories in the vertical dimension (climb, descend) in airspace. TCAS interacts with the following components: Radio Altimeter, A/C Discretes, Receiver, Transmitter, Mode-S Transponder and other flight control units [33]. TCAS issues two types of alerts:

- Traffic Advisories (TAs) to assist the pilot in the visual search for the intruder aircraft and to prepare the pilot for a potential resolution advisory.
- Resolution Advisories (RAs) to recommend maneuvers that will either increase or maintain the existing vertical separation from an intruder aircraft. When the intruder aircraft is also fitted with TCAS, both TCAS systems co-ordinate their RAs through the Mode S data link to ensure that complementary RAs are selected. If the intruder aircraft is equipped with Mode A transponder only, TCAS provides just the TA. If the intruder aircraft is equipped with Mode C or Mode S transponder only, TCAS provides TA and RA to the pilot.

2.3. AUTOMATIC DEPENDENT SURVEILLANCE BROADCAST (ADS-B)

ADS-B is a replacement for (or supplement to) traditional radar based surveillance of aircraft. ADS-B uses satellite-based navigation systems to determine an aircraft's precise location in space. The system then converts the position into a digital code, which is combined with other information such as the type of aircraft, flight number, speed, and intent. An ADS-B equipped aircraft broadcasts its information through an omnidirectional fashion, and any aircraft or ATC facility can receive this information (See Figure 2.3). These broadcasts are not in response to interrogations, unlike existing transponder technology. ADS-B transmission occurs at much lower rate than SSR replies. Note that ADS-B cannot

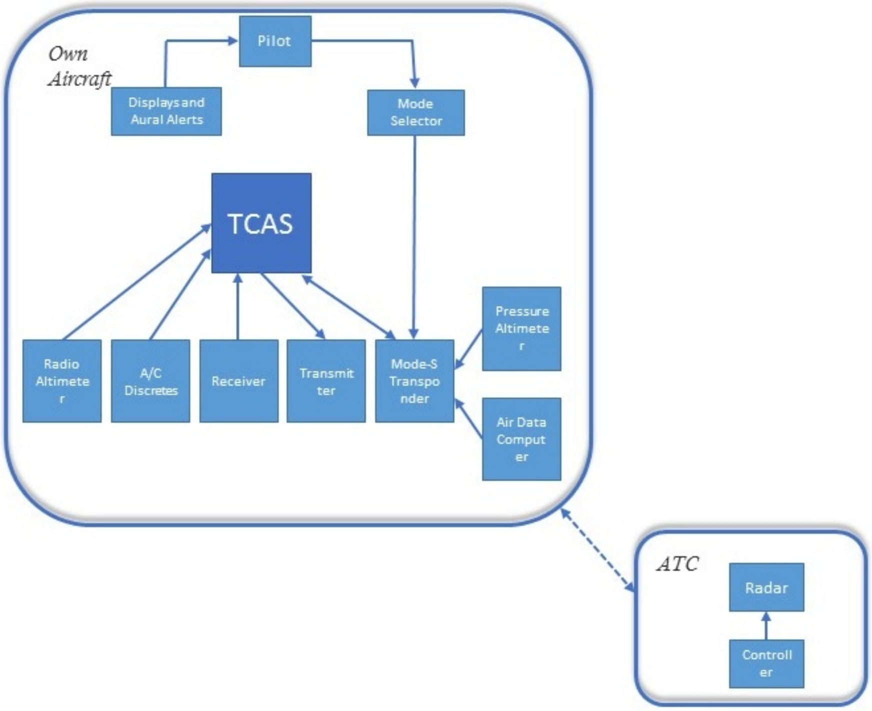

Figure 2.1. TCAS II Interacting Entities

replace existing SSRs until all aircraft are equipped with ADS-B equipment to broadcast state vector information. ADS-B will most likely be mandated in airspace where Mode-C transponders are currently required [34].

2.4. PITOT STATIC SYSTEM

A pitot-static system is a system of pressure-sensitive instruments that is most often used in aviation to determine an aircraft's airspeed, Mach number, altitude, and altitude trend (See Figure 2.4). A pitot-static system generally consists of a pitot tube, a static port, and the pitot-static instruments [35]. This equipment is used to measure the forces acting on a vehicle as a function of the temperature, density, pressure and viscosity of

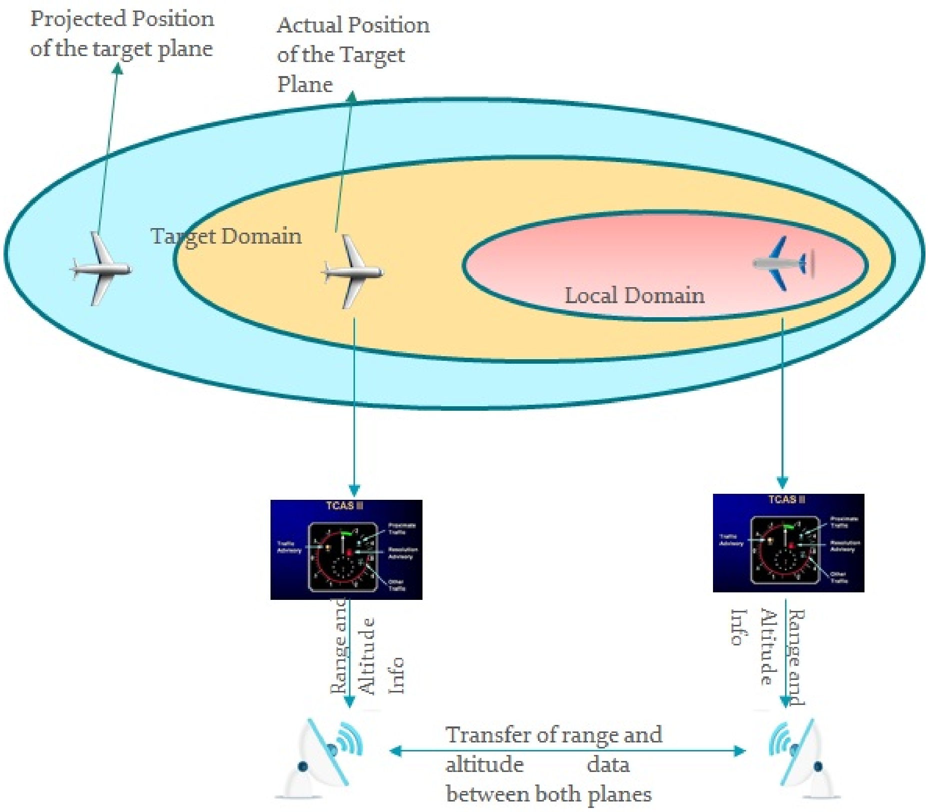

Figure 2.2. Communication Between Two TCAS Equipped Aircrafts

the fluid in which it is operating. Other instruments that might be connected are air data computers, flight data recorders, altitude encoders, cabin pressurization controllers, and various airspeed switches [36].

Managing a static system malfunction requires that the pilot know and understand the airplane's pitot-static system. If a system malfunction is suspected, the pilot should confirm it by opening the alternate static source. It is a source of ambient air pressure from the depressurized area within an aircraft for use when the static vent malfunctions. It is less accurate, but usable in emergency situations. This should be done while the airplane is climbing or descending.

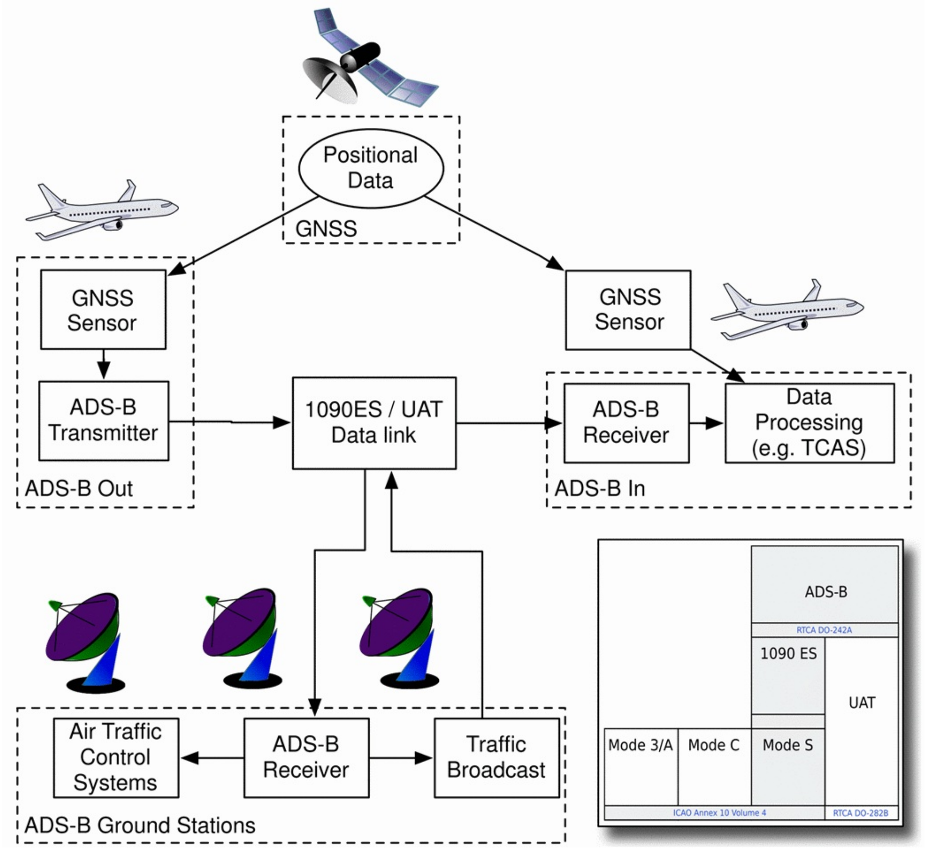

Figure 2.3. ADS-B System

2.5. LIFT RESERVE INDICATOR (LRI)

An LRI is a system used to measure the amount of lift being produced by a wing in any given situation, by comparing the static pressure to the dynamic pressure in a probe mounted in a fixed position under the wing. It is used to maintain a safe margin from a stalling condition, and used in takeoff, landing, and while maneuvering at any attitude or angle. When displayed on a simple pressure differential gauge with a modified face, the pilot knows the amount of lift reserve available at any given moment [37].

The LRI integrates both airspeed and angle of attack in a single readout reliably and continuously displaying an aircraft's margin over stall despite the wide range of variables to which an aircraft is subject. This helps the pilot to maintain proper course in case of incorrect airspeed readings.

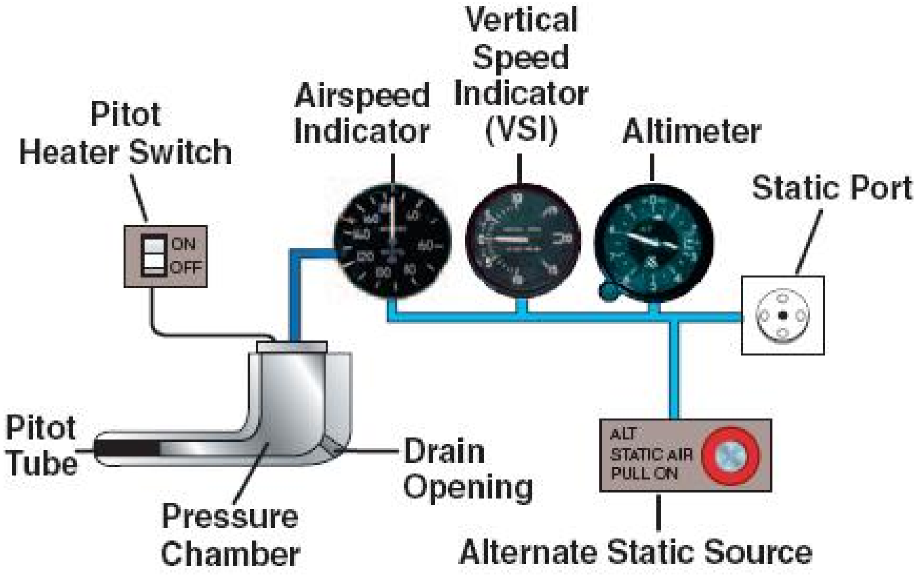

Figure 2.4. Pitot Static System [1]

2.6. INERTIAL NAVIGATION SYSTEM (INS)

An INS is a totally self-contained dead reckoning system [38]. Dead reckoning is the process of calculating current position of the aircraft by using a previously determined position, or fix, and advancing that position based upon known or estimated speeds over elapsed time and course. Given its starting position, INS keeps track of all movements in all directions so it calculates the aircraft's flight position in relation to that point. To detect movement, the INS uses three accelerometers: one north-south, one east-west, and one up-down mounted on a stable platform. The platform is stabilized using three gyros, one each for pitch, yaw and roll. This way the aircraft's movement is constantly monitored and helps the pilot keep the aircraft on course.

Advanced INS use ring laser gyros that are made up of a series of lasers aligned in the same plane and forming a ring. Interference patterns are generated as the aircraft accelerates indicating changes in the airplane's movement.

The INS must be initialized on the ramp prior to takeoff. The pilot merely enters the aircraft's coordinates and the system performs the calculations since it has an internal clock calendar. This system computes track, drift angle, cross track error, distance traveled, distance remaining and flight time remaining for the pilot.

The INS system uses a multitude of invariant data such as measurements provided by accelerometers and gyroscopes to track the position and orientation of an object relative to a known starting point, orientation and velocity.

3. PROBLEM STATEMENT

Aviation security is at the forefront of society, mostly as a protection against terrorism and national security threats in the physical world. The introduction of cyber advances and tight cyber-physical integration within aircraft, however, raises new aviation security considerations for threats from and to cyberspace. Cyberspace has well-known vulnerabilities to physical world exploits such as radio jamming and equipment compromise. The growth of future aviation, hence, heavily weighs on understanding cyber-physical threats to aircraft, identifying new threats from cyber-physical integration, and managing security risks. Factors contributing to risk include system malfunction, intruders, mistakes by the pilots and environmental factors.

The challenges continue to grow as individual systems evolve, operate with greater autonomy and intelligence, and operate as part of a networked system of systems. A more concise way to classify cyber and physical assets in aviation and the information flow paths between these assets, which must be protected against threats and from becoming a threat is needed. Additionally, a more formal way of representing various attacks and possible mitigation measures is needed in order to ensure security of the entire system.

Air-to-air, air-to-ground, and satellite-to-air communications were considered to perform security analysis either from the cyber point of view or the physical point of view. Certainly a formal method is required to perform security analysis considering the cyber-physical interactions, looking for the potential risks and changes that have been occurring in airplane systems, along with the implications of those changes.

This thesis is aimed at addressing the above specified issues by focusing on cyber and physically enabled attacks by using a model-based approach to identify security risks and provide mitigation measures.

4. RELATED WORK

This work mainly focuses on nondeducibility in multiple security domains involving commands from multiple sources and applies to cyber physical systems involving complex interactions between different states of the system.

It is well known in the aviation community that the ATC system, which is currently being rolled out, called automatic dependent surveillance-broadcast (ADS-B), had not been developed with security in mind and is susceptible to a number of different radio frequency (RF) attacks. The problem has recently been widely reported in the press [39, 40, 41, 42] and at hacker conventions [43, 44, 45]. Academic researchers, too, proved the ease of compromising the security of ADS-B with current off-the-shelf hard- and software [46]. This broad news exposure led the International Civil Aviation Organization (ICAO) to put the security of civil aviation on the agenda of the 12th air navigation conference, identifying "cyber security as a high level impediment to implementation that should be considered as part of the roadmap development process" and creating a task force to help with the future coordination of the efforts of involved stakeholders.

Similar work has been carried out in [24] but this work is limited to accepting commands from a single source and it does not cover all the possible state transitions in the system. Similar work has been carried out in [47] which used MSDND to model the security of a chemical plant using BIT logic. The major drawback is the inability to represent the state transitions and interactions in a concise way and our work covers this by segregating the system into multiple security domains. These security domains contains multiple states and the security issues between the state interactions are considered.

5. MULTIPLE SECURITY DOMAIN NONDEDUCIBILITY

5.1. MODAL LOGIC MODEL

Formally, a set of worlds W are defined, consisting of distinct worlds, w_0,w_1,...,w_n where, if m state variables are present, S_1,S_2,...,S_m then it is possible to have 2^m distinct worlds.

The worlds are connected by a set of transitions, $\{wRw^{'}\}$. Changing any state variable causes a transition from the current world, w, to another world, $w^{'}$ where all other state variables retain their values. Together, the set of worlds and transitions define a frame, $F = \{W,R\}$.

A set of valuation functions are defined, $\{V\}$, such that $V^i_{s_x}(w)$ returns the value of state variable s_x as seen by an entity i in world w. **NOTE:** If no valuation function exists to return the value of a state variable, say s_i, then the model can never determine the value of that state variable nor the value of any logical expression dependent upon that state variable. By combining the valuation functions and the frame, the model can be defined as $M = \{F,V\}$ or $M = \{W,R,V\}$ [24].

5.2. MULTIPLE SECURITY DOMAIN NONDEDUCIBILITY MODEL

Extending existing models to multiple security domains is problematic. An extended version of the Multiple Security Domains Nondeducibility Model using statecharts and local invariants present in the CPS was proposed in this thesis. An entity i is defined as any part of the system capable of independent observation or action.

The Event System (ES) can be divided into multiple security domains, SD^i, as viewed by each entity i in the model. These domains may, or may not, overlap with each other. These multiple security domains conform to the following rules:

$$\cup_{i\in I} SD^i = (ES) \tag{5.1}$$

5.3. 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.

$$f(S_a, S_b, S_c, ...) = \begin{cases} \textit{where exactly one of } S_a, S_b, S_c, ... \textit{ is True} \\ \textit{otherwise False} \end{cases} \tag{5.2}$$

5.4. DEFINITION: MULTIPLE SECURITY DOMAIN NONDEDUCIBILITY

There exists some world with multiple states in which at any instance one state is true and the others are false, but an entity i has no valuation function for those states. In security domain SD^i, the states cannot be evaluated to either true or false.

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

Note: There exists a valuation function if all the state variables return true and there does not exist a valuation function if any of the state variables returns false.

5.5. LOCAL INVARIANTS

An invariant is a quantity that remains unchanged under certain classes of transformations. Invariants are extremely useful for classifying mathematical objects because they usually reflect intrinsic properties of the object of study. Local invariants are the invariants between any pair of attributes of the system.

5.6. DEFINITION: DF FUNCTION

In the axiomatic view, *df* tests whether an event variable has a defined value [48]. This function contains series of transitions which results in an output

$$df\_ : P(eventX) \tag{5.4}$$

For example, a label expression which requires a signal a and generates a signal b in the next step is written as dfa/dfb'.

Note: df is different from valuation function which returns the value of a state variable seen by an entity.

5.7. PROVERIF WITH RESPECT TO MSDND

ProVerif is capable of proving reachability properties, correspondence assertions, and observational equivalence. In this thesis, reachability property and observational equivalence are used to prove deducibility.

5.7.1. Reachability Property. Given a system and a property p, reachability model checking is based on an exhaustive exploration of the reachable state space of the system, testing whether there exists a state where p holds. The main obstacle to this approach is the state-explosion problem reflecting the fact that the system's state space is often prohibitively large to be entirely explored. Abstractions have been proven a useful tool in coping with state explosion. Model checking using abstractions consists of exploring a abstract state space rather than the concrete one.

This thesis abstracts the collision region of the TCAS and ADS-B systems into a 9-grid system in which each grid cell is considered to be of same size. Each grid cell is denoted as XY_Coord_Node_x where x $\in$ {1,2,3,4,5,6,7,8,9}. If both planes are in different grid cells, they are equidistant from each other. For e.g., if the plane-1 is in XY_Coord_Node_1 and plane-2 is in XY_Coord_Node_9, they are equidistant from each other (See Figure 5.1). If both planes are in the same grid cell, collision is bound to happen. If they are in different grid cells, the pilots can follow the RAs suggested by the TCAS system and avoid collision. ProVerif attempts to prove that a state in which the nodes are known to the adversary is unreachable (that is, it tests the query *not attacker(XY_Coord)*, and this query is true when the location is not derivable by the adversary). This makes ProVerif suitable for proving the secrecy of data with respect to MSDND.

If ProVerif's output is of the form RESULT not attacker:(XY_Coord[]) is true, the attacker has not been able to obtain the location XY_Coord. The attacker has, however, been able to obtain the location XY_Coord as denoted by the RESULT not attacker:(XY_Coord[]) is false. It follows that when ProVerif is supplied with *query attacker(M).*, internally ProVerif attempts to show *not attacker(M)* and hence *RESULT not attacker(M) is true.* means that the secrecy of *M* is preserved by the protocol.

5.7.2. Observational Equivalence. The MSDND model uses the most general class of equivalences $P \approx Q$ where the processes P and Q have the same structure and differ only in the choice of terms. These equivalences are written in ProVerif by a single "biprocess" that encodes both P and Q. Such a biprocess uses the construct *choice[M,M']* to represent the terms that differ between P and Q: P uses the first component of the

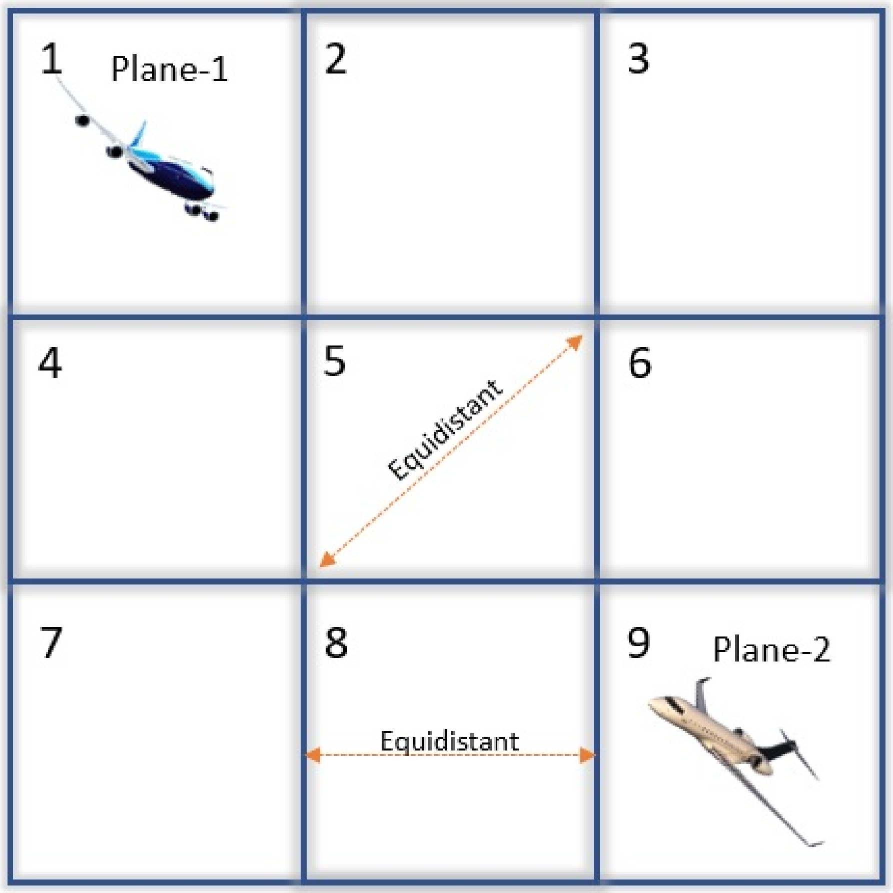

Figure 5.1. 9-Grid System

choice, *M*, while *Q* uses the second one, *M'*. The MSDND model uses a similar approach to identify whether a state is false or true, considering parallel composition of different security domains.

In order to correctly understand the results of the observational equivalence property in ProVerif, it is important to understand the difference between the attack derivation and the attack trace. The attack derivation is an explanation of the actions that the attacker has to execute in order to break the security property, in the internal representation of ProVerif.

Because this internal representation uses abstractions, the derivation is not always executable in reality; for instance, it may require the repetition of certain actions that can in fact never be repeated, for instance because they are not under a replication. In contrast, the attack trace refers to the semantics of the applied pi calculus, and always corresponds to an executable trace of the considered process. ProVerif can display three kinds of results:

- RESULT [Query] is true: The query is proved, there is no attack. In this case, ProVerif displays no attack derivation and no attack trace.
- RESULT [Query] is false: The query is false, ProVerif has discovered an attack against the desired security property.
- RESULT [Query] cannot be proved: This is a "don't know" answer. ProVerif could not prove that the query is true and also could not find an attack that proves that the query is false. Since the problem of verifying protocols for an unbounded number of sessions is undecidable, this situation is unavoidable. Still, ProVerif gives some additional information that can be useful in order to determine whether the query is true. In particular, ProVerif displays an attack derivation. By manually inspecting the derivation, it is sometimes possible to reconstruct an attack. For observational equivalence properties, it may also display an attack trace, even if this trace does not prove that the observational equivalence does not hold.

6. MODAL LOGIC WITH THE USE OF INVARIANTS

This section presents an overview of the vulnerabilities associated with air traffic surveillance systems. In order to do this, the scenarios are divided into two parts: 1) identifying the compromised system or a malfunctioning system. 2) using the invariants associated with air traffic surveillance systems and applying those invariants to the MSDND model which helps in identifying the vulnerabilities associated with the system.

6.1. RADAR TRANSPONDER FAILURE

This section presents two different scenarios to check if the compromised radar transponder can be identified by the pilot.

6.1.1. Scenario 1: Without Invariants. In case of a failure in the radar transponder, the MSDND model yields Nondeducibility, thereby stopping critical information flow to the pilot and the ATC controller.

Proof: Let us consider the possibility of failure in the radar transponder of the aircraft (See Figure 6.1). The ATC controller receives incorrect information about the aircraft identification. This could lead to improper communication between the ATC controller and the pilot. The controller and the pilot cannot distinguish the source of incorrect information.

Table 6.1 presents the set of logical conditions, φ_i, p, c, t, r, x that can be evaluated to determine the interactions between the aircraft and the ATC.

Once the flight information is retrieved, the ATC controller and the pilot can observe that there is a mismatch between the flight data and ATC data. The ATC controller and the pilot cannot distinguish whether there is a failure in the aircraft transponder or the radar system of ATC. The pilot can sense the position of the aircraft from the altitude reading, but cannot evaluate the source of incorrect information being sent to the ATC and TCAS.

Table 6.1. Radar Transponder Logical Conditions And States - Without Invariants

φ_i	**States**	**Functions**
φ_0	Normal	Plane functions normally.
φ_1	Interrogations	Initiation of transmission of interrogations from ATC.
φ_2	Replies	Initiation of transmission of replies from XPDR.
φ_3	TCAS Data	TCAS System knows the position of the plane.
φ_4	Radar Data	ATC Radar System knows the location of the plane.
φ_5	Communication	Initiation of communication between pilot and controller based on TCAS Data and Radar Data respectively.
S_p	p = T	$\boldsymbol{p = \neg\varphi_3 \wedge \neg\varphi_5}$ ~df(TCAS data) ∧ ~df(communication) Output = TA or RA
S_c	c = T	$\boldsymbol{c = \neg\varphi_4 \wedge \neg\varphi_5}$ ~df(Radar) ∧ ~df(communication) Output = Flight Location
S_t	t = T	$\boldsymbol{t = \neg\varphi_2 \wedge \neg\varphi_3}$ ~df(replies) ∧ ~df(TCAS data) Output = Position

S_r	r = T	$\boldsymbol{r = \varphi_1 \wedge \neg\varphi_2 \wedge \neg\varphi_4}$
		df(interrogations) ∧ ~df(replies) ∧ ~df(Radar data)
		Output = Display

S_x	x = T	$\boldsymbol{x = \varphi_1 \wedge \neg\varphi_2}$ df(interrogations) ∧ ~df(replies)

The two security domains in this scenario are SD^P {pilot domain} and SD^C {ATC controller domain}. By combining the valuation functions in SD^P and SD^C,

$$S_p = \neg\varphi_3 \wedge \neg\varphi_5 \Rightarrow \nexists V^P_{\sim t} \tag{6.1}$$

Since the information received from the pilot domain is faulty, the pilot cannot valuate for correctness of the TCAS data in that domain.

$$S_c = \neg\varphi_4 \wedge \neg\varphi_5 \Rightarrow \nexists V^C_{t} \tag{6.2}$$

Since the information received from the pilot domain is faulty, the ATC controller cannot valuate for correctness of the TCAS data in that domain.

By combining Equation 6.1 and Equation 6.2,

$$MSDND(ES) = \exists w \in W : [w \vdash \Box f(S_p, S_c)] \wedge [w \models (\nexists V^P_{\sim t} \wedge \nexists V^C_t] \tag{6.3}$$

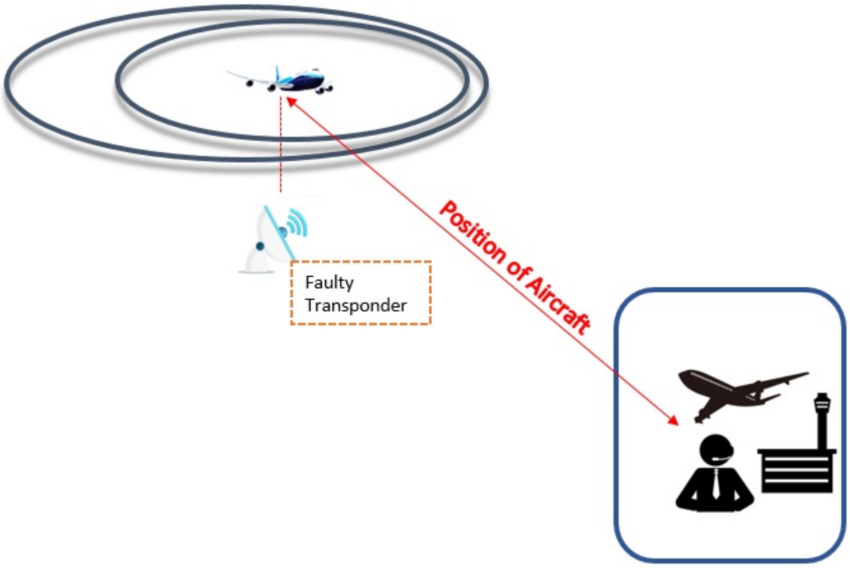

Figure 6.1. Faulty Radar Transponder

Therefore, from physical observation the ATC controller can deduce that something is going wrong but cannot deduce the source that is sending incorrect information to the TCAS and ATC radar.

Hence, the system is *Nondeducible secure* to the pilot and the ATC controller as they can deduce that something is going wrong, but cannot deduce the entity responsible for transmitting incorrect data.

6.1.2. Scenario 2: Using Invariants. In case of radar transponder compromise, the MSDND model yields deducibility, thereby allowing critical information flow to the pilot and the ATC controller.

Proof: The above mentioned scenario can be made deducible to the pilot and ATC controller by relying on alternate information flow path from INS, which helps in identifying the compromised system (See Figure 6.2). Hence, the pilot and the ATC controller can deduce the source of incorrect information based on the invariant data computed by INS.

In case of failure in the radar transponder, the pilot can make use of the INS system. The velocity is calculated using

$$v = v_0 + at \tag{6.4}$$

where, v_0 is the initial velocity, v is the final velocity, a is the acceleration and t is the time between observations.

Using the final velocity from Equation 6.4, the distance travelled by the aircraft can be calculated using

$$d = v * t \tag{6.5}$$

where, d is the distance travelled, v is the velocity and t is the time.

Note: In Table 6.2, d is the distance travelled by the plane projected on the INS system.

Table 6.2 presents the set of logical conditions, $\varphi_i, p, c, t, r, x, i$ that can be evaluated to determine the interactions between the aircraft and the ATC. The two security domains in this scenario are SD^T {TCAS domain} and SD^I {INS domain}. By combining the valuation functions in SD^T and SD^I with respect to invariants from Equation 6.4 and Equation 6.5 in the pilot's domain,

$$S_t = \neg\varphi_2 \wedge \neg\varphi_3 \Rightarrow \nexists V^P_{\sim i} \tag{6.6}$$

Since the information received from the pilot domain is faulty, the pilot cannot valuate for correctness of the TCAS data in that domain.

$$S_i = \varphi_6 \wedge \varphi_7 \Rightarrow \nexists V^P_i \tag{6.7}$$

Table 6.2. Radar Transponder Logical Conditions And States - Using Invariants

φ_i	**States**	**Functions**
φ_0	Normal	Plane functions normally.
φ_1	Interrogations	Initiation of transmission of interrogations from ATC.
φ_2	Replies	Initiation of transmission of replies from XPDR.
φ_3	TCAS Data	TCAS System knows the position of the plane.
φ_4	Radar Data	ATC Radar System knows the location of the plane.
φ_5	Communication	Correct communication between pilot and controller based on TCAS Data and Radar Data respectively.
φ_6	d	Pilot checks the position of the plane using INS and verifies it with the TCAS data.
φ_7	Verification	Pilot and the controller checks and verifies the position of the respective aircraft and verifies it with the position data displayed.
S_p	p = T	$\boldsymbol{p = \neg\varphi_3 \land \neg\varphi_5 \land \varphi_6 \land \varphi_7}$ ~df(TCAS data) ∧ ~df(communication) ∧ df(d) ∧ df(Verification) Output = TA or RA
S_c	c = T	$\boldsymbol{c = \neg\varphi_4 \land \neg\varphi_5 \land \varphi_7}$ ~df(Radar) ∧ ~df(communication) ∧ df(Verification) Output = Flight Location
S_t	t = T	$\boldsymbol{t = \neg\varphi_2 \land \neg\varphi_3}$ ~df(replies) ∧ ~df(TCAS data) Output = Position

S_r	r = T	$\boldsymbol{r = \varphi_1 \land \neg\varphi_2 \land \neg\varphi_4}$
		df(interrogations) ∧ ~df(replies) ∧ ~df(Radar data)
		Output = Display

S_x	x = T	$\boldsymbol{x = \varphi_1 \land \neg\varphi_2}$ df(interrogations) ∧ ~df(replies)
S_i	i = T	$\boldsymbol{i = \varphi_6 \land \varphi_7}$ df(d) ∧ ~df(Verification)

Since the information received from the pilot domain is faulty, the ATC controller cannot valuate for correctness of the TCAS data in that domain.

By combining Equation 6.6 and Equation 6.7,

$$MSDND(ES) = \exists w \in W : [w \vdash \Box f(S_t, S_i)] \wedge [w \models (\nexists V^{P}_{\sim i} \wedge \exists V_i^P] \qquad (6.8)$$

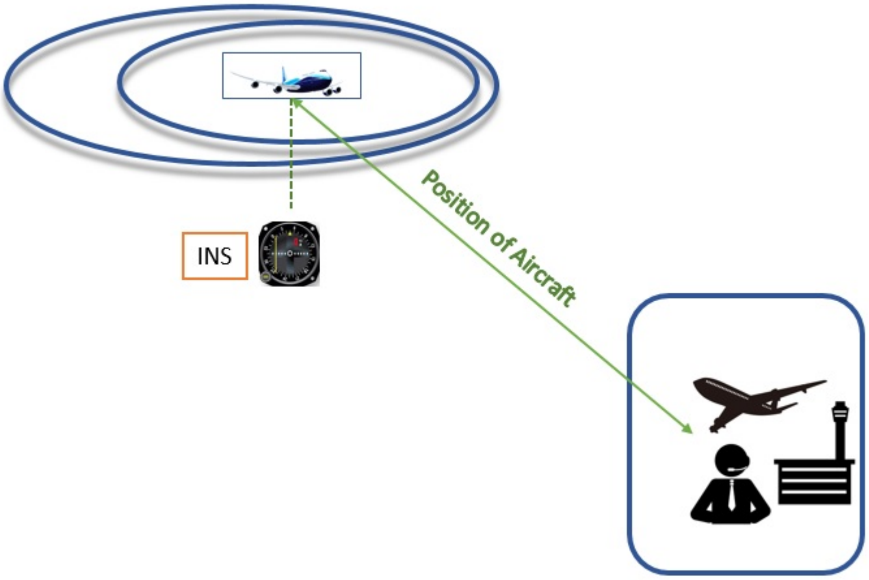

Figure 6.2. INS Helps Identify The Faulty Radar Transponder

Hence, the system is *not Nondeducible secure* to the pilot and the ATC controller as they can deduce that TCAS system is responsible for transmitting incorrect data.

The ProVerif code in APPENDIX A proves that this attack is *not Nondeducible secure* when INS is used as an alternate information flow path. As can be interpreted from "RESULT not attacker(XY_Coord[]) is false", the TCAS process is compromised.

6.2. RADAR TRANSPONDER FAILURE AND TCAS

This section presents two different scenarios to check if the compromised radar transponder can be identified by the pilot using TCAS.

6.2.1. Scenario 1: Without Using Invariants. In case of radar transponder compromise of plane-1, the MSDND model yields Nondeducibility, thereby stopping critical information flow to the pilots and the ATC Controller.

Proof: It is a known fact that most of the world is not covered by radar. Part of that is a technical challenge. The air traffic system runs on ground-based radar, and most of the world is covered in water. Ground stations are not available in the ocean. For that reason, once the flight is more than a few miles off the coast, it is going to be out of radar range and this is called no-radar zone.

In this case, the ATC controller cannot receive information about the aircraft's identification. This could lead to miscommunication between both the pilots in case of any failure. Let us consider the possibility of malfunction of the TCAS-1 system. This leads to the display of incorrect data on the TCAS system and the pilots cannot distinguish which TCAS system is faulty. But the controller cannot identify the plane with a faulty TCAS system from the radar data and cannot provide necessary maneuvers to avoid collision by directing the pilots in correct directions (See Figure 6.3).

Table 6.3 presents the set of logical conditions, $\varphi_i, p_1, p_2, c, t_1, t_2, r$ that can be evaluated to determine the interactions between the planes and the ATC.

In case the plane is in a no-radar zone, once the flight information is retrieved, pilots communicate with each other in case of close proximity to avoid collision. If the TCAS system of pilot-1 presents wrong information due to technical failure, the pilots cannot distinguish which plane is presenting wrong information.

The two security domains in this scenario are SD^P {pilot domain} and SD^C {ATC controller domain}. By combining the valuation functions in SD^P and SD^C,

$$S_p = \neg\varphi_2 \wedge \varphi_3 \wedge \neg\varphi_4 \wedge \varphi_5 \wedge \neg\varphi_7 \Rightarrow \nexists V^P_{\sim t} \tag{6.9}$$

Table 6.3. Radar And TCAS Logical Conditions And States - Without Invariants

φ_i	States	Functions
φ_0	Normal	Plane functions normally.
φ_1	Interrogations	Correct transmission of interrogations from ATC.
φ_2	Reply-1	Correct transmission of replies from XPDR-1.
φ_3	Reply-2	Correct transmission of replies from XPDR-2.
φ_4	TCAS-1 Data	TCAS-1 knows the position of the plane-1 and plane-2.
φ_5	TCAS-2 Data	TCAS-2 knows the position of the plane-1 and plane-2.
φ_6	Radar Data	ATC Radar System knows the location of the plane-1 and plane-2.
φ_7	Communication	Correct communication between pilot-1, pilot-2 and the Controller from TCAS Data and Radar Data.
S_{p1}	p1 = T	$\boldsymbol{p1 = \neg\varphi_2 \wedge \neg\varphi_4 \wedge \neg\varphi_7}$ ~df(Reply-1) ∧ ~df(TCAS-1 data) ∧ ~df(communication) Output = TA or RA

S_{p2}	p2 = T	$\boldsymbol{p2 = \varphi_3 \wedge \varphi_5 \wedge \neg\varphi_7}$
		df(Reply-2) ∧ df(TCAS-2 data) ∧ ~df(communication)
		Output = TA or RA

S_c	c = T	$\boldsymbol{c = \neg\varphi_2 \wedge \varphi_3 \wedge \neg\varphi_6 \wedge \neg\varphi_7}$ ~df(Reply-1) ∧ df(Reply-2) ∧ ~df(Radar Data) ∧ ~df(communication) Output = Flight Location
S_{t1}	t1 = T	$\boldsymbol{t1 = \neg\varphi_2 \wedge \neg\varphi_4}$ ~df(Reply-1) ∧ ~df(TCAS-1 data) Output = Position

S_{t2}	t2 = T	$\boldsymbol{t2 = \varphi_3 \wedge \varphi_5}$
		df(Reply-2) ∧ df(TCAS-2 data)
		Output = Position

S_r	r = T	$\boldsymbol{r = \varphi_1 \wedge \neg\varphi_2 \wedge \varphi_3 \wedge \neg\varphi_6}$
		df(Interrogations) ∧ ~df(Reply-1) ∧ df(Reply-2) ∧ ~df(Radar data)
		Output = Display

Since the information received from the pilot's domain is faulty, the pilots cannot valuate for correctness of the TCAS data in that domain.

$$S_c = \neg\varphi_2 \wedge \varphi_3 \wedge \neg\varphi_6 \wedge \neg\varphi_7 \Rightarrow \exists V_t^C \tag{6.10}$$

Since the information received from the pilot domain is faulty, the ATC controller cannot valuate for correctness of the TCAS data in that domain.

By combining Equation 6.9 and Equation 6.10,

$$MSDND(ES) = \exists w \in W : [w \vdash \Box f(S_p, S_c)] \wedge [w \models (\nexists V^P_{\sim t} \wedge \exists V^C_t] \tag{6.11}$$

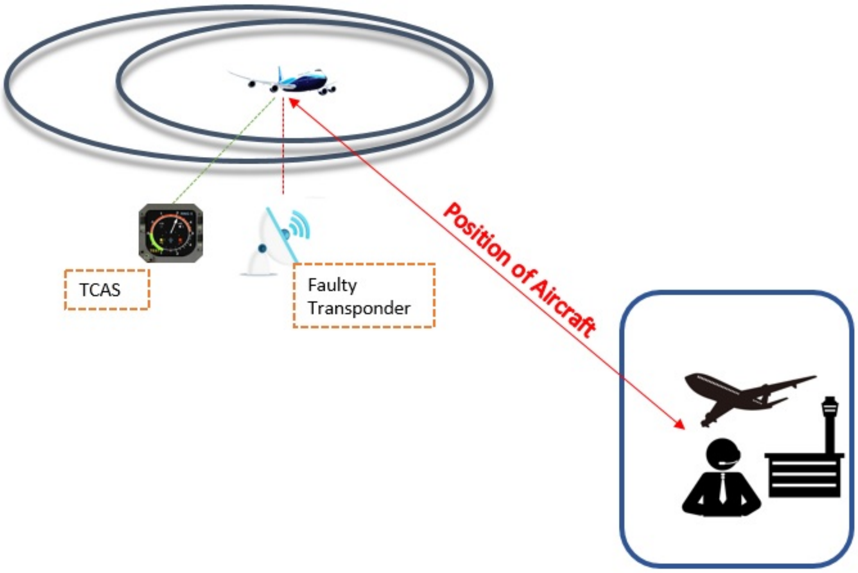

Figure 6.3. TCAS And Faulty Radar Transponder

Hence, the system is *Nondeducible* to the pilots and *not Nondeducible* to the Controller as he can deduce that something is going wrong and can determine which entity is responsible for displaying the incorrect data. This situation gives rise to dangerous situations as the pilots believe in the false data and might result in a collision of the planes.

6.2.2. Scenario 2: Using Invariants. In case of radar transponder compromise of plane-1, but using INS, the MSDND model yields deducibility, thereby allowing critical information flow to the pilots and the ATC Controller.

Proof: The above mentioned scenario can be made deducible to the pilots and ATC controller by relying on alternate information flow path from INS, which helps in identifying the compromised system. Hence, the pilot and the ATC controller can deduce the source of incorrect information based on the invariant data computed by INS (See Figure 6.4).

In case of failure in the radar transponder, the pilot can make use of the INS system. The velocity is calculated using Equation 6.4.

Using the final velocity, distance travelled by the aircraft is calculated using Equation 6.5.

Note: In Table 6.4, d is the distance travelled by the plane projected on INS system.

Table 6.4 presents the set of logical conditions, $\varphi_i, p_1, p_2, c, t_1, t_2, r, i1, i2$ that can be evaluated to determine the interactions between the planes and the ATC.

The two security domains in this scenario are SD^T {TCAS domain} and SD^I {INS domain}. By combining the valuation functions in SD^T and SD^I with respect to invariants from Equation 6.4 and Equation 6.5 in pilot-1 domain,

$$S_t = \neg\varphi_2 \wedge \varphi_3 \wedge \neg\varphi_4 \wedge \varphi_5 \Rightarrow \nexists V^P_{\sim i} \tag{6.12}$$

Since the information received from the TCAS domain is faulty, the pilot cannot valuate for correctness of the TCAS data in that domain.

$$S_i = \varphi_8 \wedge \varphi_9 \wedge \varphi_{10} \Rightarrow \exists V^P_i \tag{6.13}$$

Since the information received from the INS domain is not faulty, the pilot can valuate for correctness of the TCAS data in that domain.

By combining Equation 6.12 and Equation 6.13,

$$MSDND(ES) = \exists w \in W\colon\ [w \vdash \Box f(S_t, S_i)] \wedge [w \models (\nexists V^P_{\sim i} \wedge \exists V^P_i]\tag{6.14}$$

Table 6.4. Radar And TCAS Logical Conditions And States - Using Invariants

φ_i	**States**	**Functions**
φ_0	Normal	Plane functions normally.
φ_1	Interrogations	Correct transmission of interrogations from ATC.
φ_2	Reply-1	Correct transmission of replies from XPDR-1.
φ_3	Reply-2	Correct transmission of replies from XPDR-2.
φ_4	TCAS-1 Data	TCAS-1 knows the position of the plane-2.
φ_5	TCAS-2 Data	TCAS-2 knows the position of the plane-1.
φ_6	Radar Data	ATC Radar System knows the location of the plane-1 and plane-2.
φ_7	Communication	Correct communication between pilot-1, pilot-2 and the Controller from TCAS Data and Radar Data.
φ_8	d_1	Pilot-1 checks the position of the plane using INS and verifies it with the TCAS data.
φ_9	d_2	Pilot-2 checks the position of the plane using INS and verifies it with the TCAS data.
φ_{10}	Verification	Pilot-1, pilot-2 and the controller checks and verifies the INS position data with respect to TCAS position data.
S_{p1}	p1 = T	**$p1 = \neg\varphi_2 \wedge \neg\varphi_4 \wedge \neg\varphi_7$** ~df(Reply-1) ∧ ~df(TCAS-1 data) ∧ ~df(communication) Output = TA or RA

S_{p2}	p2 = T	**$p2 = \varphi_3 \wedge \varphi_5 \wedge \neg\varphi_7$**
		df(Reply-2) ∧ df(TCAS-2 data) ∧ ~df(communication)
		Output = TA or RA

S_c	c = T	**$c = \neg\varphi_2 \wedge \varphi_3 \wedge \neg\varphi_6 \wedge \neg\varphi_7$** ~df(Reply-1) ∧ df(Reply-2) ∧ ~df(Radar Data) ∧ ~df(communication) Output = Flight Location
S_{t1}	t1 = T	**$t1 = \neg\varphi_2 \wedge \neg\varphi_4$** ~df(Reply-1) ∧ ~df(TCAS-1 data) Output = Position

S_{t2}	t2 = T	**$t2 = \varphi_3 \wedge \varphi_5$**
		df(Reply-2) ∧ df(TCAS-2 data)
		Output = Position

S_r	r = T	**$r = \varphi_1 \wedge \neg\varphi_2 \wedge \varphi_3 \wedge \neg\varphi_6$**
		df(Interrogations) ∧ ~df(Reply-1) ∧ df(Reply-2) ∧ ~df(Radar data)
		Output = Display

S_{i1}	i1 = T	**$i1 = \varphi_8 \wedge \varphi_{10}$** df(d1) ∧ df(Verification)
S_{i2}	i2 = T	**$i2 = \varphi_9 \wedge \varphi_{10}$** df(d1) ∧ df(Verification)

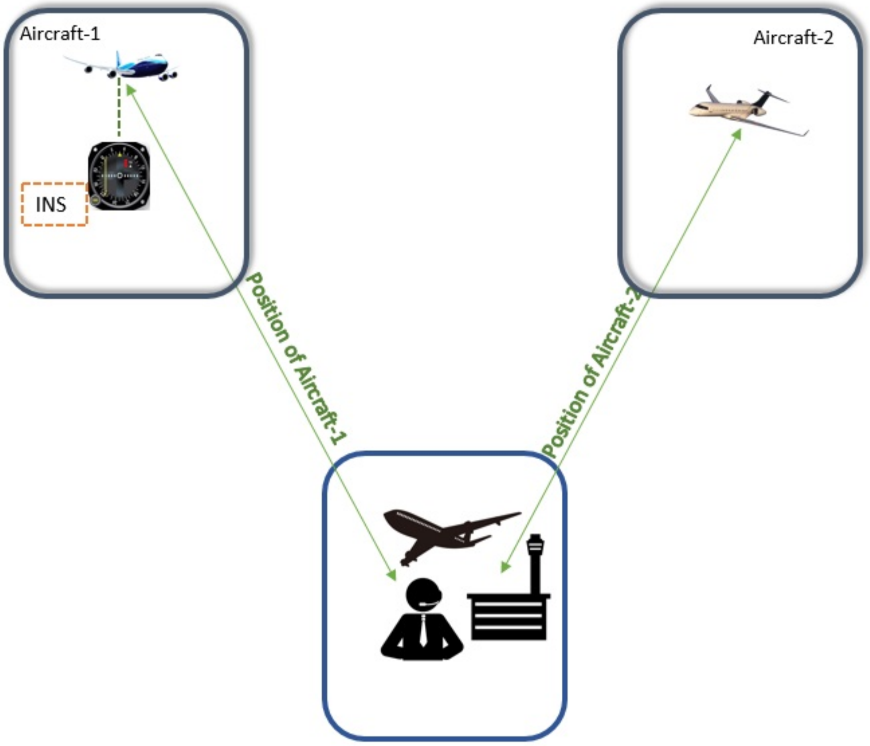

Figure 6.4. ADS-B Helps To Identify The Faulty Radar Transponder

Hence, the system is *not Nondeducible secure* to the pilots and the ATC controller as they can deduce that TCAS system is responsible for transmitting incorrect data.

The ProVerif code in APPENDIX B proves that this attack is *not Nondeducible secure* when INS is used as an alternate information flow path. As can be interpreted from "RESULT not attacker(XY_Coord[]) is false", the TCAS-1 process is compromised.

6.3. ALTIMETER FAILURE

This section presents two different scenarios to check if the compromised altimeter can be identified by the pilot.

6.3.1. Scenario 1: Without Using Invariants. In case of a failure in the altimeter, the MSDND model yields Nondeducibility, thereby stopping critical information flow to the pilot.

Proof: Let us consider the possibility of failure in the altimeter of the aircraft. The TCAS system shows an incorrect reading about the flight position to the pilot but the ATC controller receives correct information about the flight. This could lead to miscommunication between the ATC controller and the pilot, but the controller can distinguish that there is some problem with the data from the aircraft. Hence, the controller can deduce that something is going wrong, but cannot distinguish the source of incorrect information (See Figure 6.5).

Table 6.5 presents the set of logical conditions, $\varphi_i, p, c, a, t, r, x$ that can be evaluated to determine the interactions between aircraft and ATC.

Once the flight information is retrieved, the controller and the pilot can observe that there is a mismatch between the TCAS altitude reading and ATC data. The controller and the pilot cannot distinguish whether there is a failure in the TCAS system or the altimeter. The pilot can sense the position of the aircraft due to the cyber-physical nature of the plane, but cannot evaluate what is causing the incorrect information being sent to the TCAS.

The two security domains in this scenario are SD^P {pilot domain} and SD^C {ATC controller domain}. By combining the valuation functions in SD^P and SD^C,

$$S_p = \neg\varphi_2 \wedge \varphi_3 \wedge \neg\varphi_4 \wedge \varphi_5 \wedge \neg\varphi_7 \Rightarrow \nexists V^P_{\sim a} \tag{6.15}$$

Since the information received from the pilot's domain is faulty, the pilots cannot valuate for correctness of the TCAS data in that domain.

$$S_c = \neg\varphi_2 \wedge \varphi_3 \wedge \neg\varphi_6 \wedge \neg\varphi_7 \Rightarrow \nexists V^C_a \tag{6.16}$$

Since the information received from the pilot domain is faulty, the ATC controller cannot valuate for correctness of the TCAS data in that domain.

Table 6.5. Altimeter Logical Conditions And States - Without Invariants

φ_i	States	Functions
φ_0	Normal	Plane functions normally.
φ_1	Interrogations	Initiation of transmission of interrogations from ATC.
φ_2	Replies	Initiation of transmission of replies from XPDR.
φ_3	Altimeter Data	Altimeter knows the position of the plane.
φ_4	TCAS Data	TCAS System knows the position of the plane.
φ_5	Radar Data	ATC Radar System knows the location of the plane.
φ_6	Communication	Initiation of communication between pilot and controller from TCAS Data and Radar Data respectively.
S_p	p = T	$\boldsymbol{p = \neg\varphi_3 \wedge \neg\varphi_4 \wedge \neg\varphi_6}$ ~df(Altimeter Data) ∧ ~df(TCAS Data) ∧ ~df(Communication) Output = TA or RA
S_c	c = T	$\boldsymbol{c = \neg\varphi_5 \wedge \neg\varphi_6}$ ~df(Radar Data) ∧ ~df(Communication) Output = Flight Location
S_a	a = T	$\boldsymbol{a = \neg\varphi_3 \wedge \neg\varphi_6}$ ~df(Altimeter Data) ∧ ~df(Communication) Output = Flight Location
S_t	t = T	$\boldsymbol{t = \neg\varphi_3 \wedge \neg\varphi_4}$ ~df(Replies) ∧ ~df(TCAS data) Output = Position

S_r	r = T	$\boldsymbol{r = \varphi_1 \wedge \neg\varphi_5}$
		df(Interrogations) ∧ ~df(Radar data)
		Output = Display

S_x	x = T	$\boldsymbol{x = \varphi_1 \wedge \neg\varphi_2}$ df() ∧ ~df(Replies)

By combining Equation 6.15 and Equation 6.16,

$$MSDND(ES) = \exists w \in W : [w \vdash \Box f(S_p, S_c)] \wedge [w \models (\nexists V^P_{\sim a} \wedge \nexists V^C_a] \tag{6.17}$$

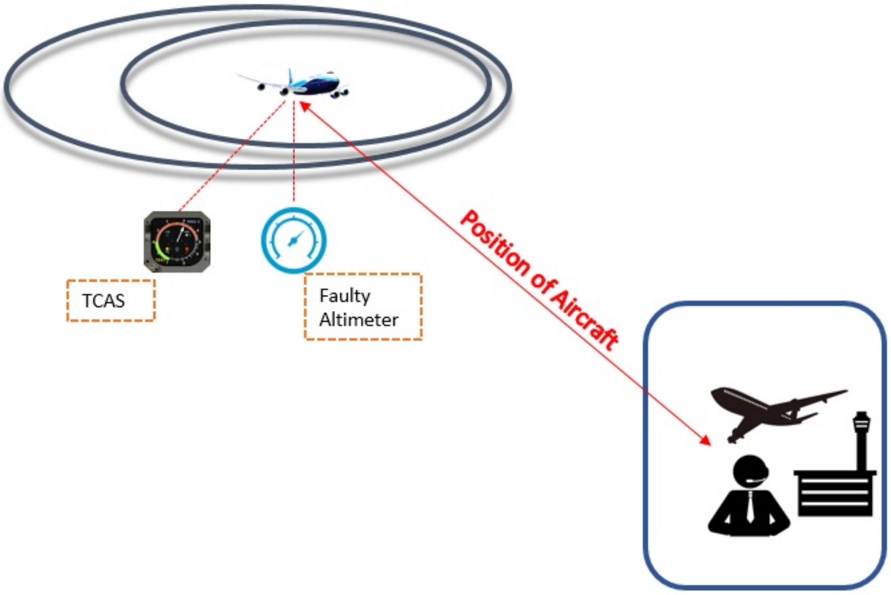

Figure 6.5. Faulty Altimeter

Therefore, from physical observation the controller can deduce that something is going wrong, but cannot deduce the source that is sending incorrect information to the TCAS, Altimeter and ATC radar.

Hence, the system is *Nondeducible secure* to the ATC controller and the pilot as they cannot deduce the source which is responsible for transmitting incorrect data.

6.3.2. Scenario 2: Using Invariants. In case of altimeter compromise, but using INS the MSDND model yields Nondeducibility, thereby allowing critical information flow to the pilot and the controller.

Proof: The above mentioned scenario can be made deducible to the pilot by relying on an alternate system; i.e., INS which helps in figuring out the compromised system. Hence, the pilot can deduce the source of incorrect information based on the invariant data computed by the INS (See Figure 6.6).

In case of failure in the altimeter, the pilot can make use of the INS system. The velocity is calculated using Equation 6.4. Using the final velocity, distance travelled by the aircraft is calculated using Equation 6.5.

Table 6.6 presents the set of logical conditions, $\varphi_i, p, c, a, t, r, i$ that can be evaluated to determine the interactions between the pilot and ATC controller.

Note: In Table 6.6, d is the distance travelled by the plane projected on INS system.

Table 6.6. Altimeter Logical Conditions And States - Using Invariants

φ_i	States	Functions
φ_0	Normal	Plane functions normally.
φ_1	Interrogations	Initiation of transmission of interrogations from ATC.
φ_2	Replies	Initiation of transmission of replies from XPDR.
φ_3	Altimeter Data	Altimeter knows the position of the plane.
φ_4	TCAS Data	TCAS System knows the position of the plane.
φ_5	Radar Data	ATC Radar System knows the location of the plane.
φ_6	Communication	Initiation of communication between pilot and controller from TCAS Data and Radar Data respectively.
φ_7	d	Pilot checks the position of the plane using INS and verifies it with the TCAS data.
φ_8	Verification	Pilot and the controller checks and verifies the INS position data with respect to TCAS position data.
S_p	p = T	$\boldsymbol{p = \neg\varphi_3 \wedge \neg\varphi_4 \wedge \neg\varphi_6 \wedge \varphi_7 \wedge \varphi_8}$ ~df(Altimeter Data) ∧ ~df(TCAS Data) ∧ ~df(Communication) ∧ df(d) ∧ df(Verification) Output = TA or RA
S_c	c = T	$\boldsymbol{c = \neg\varphi_5 \wedge \neg\varphi_6 \wedge \varphi_8}$ ~df(Radar Data) ∧ ~df(Communication) Output = Flight Location
S_a	a = T	$\boldsymbol{a = \neg\varphi_3 \wedge \neg\varphi_6}$ ~df(Altimeter Data) ∧ ~df(Communication) Output = Flight Location
S_t	t = T	$\boldsymbol{t = \neg\varphi_3 \wedge \neg\varphi_4}$ ~df(Replies) ∧ ~df(TCAS data) Output = Position

S_r	r = T	$\boldsymbol{r = \varphi_1 \wedge \neg\varphi_5}$
		df(Interrogations) ∧ ~df(Radar data)
		Output = Display

S_i	i = T	$\boldsymbol{i = \varphi_7 \wedge \varphi_8}$ df(d) ∧ df(Verification)

The two security domains in this scenario are SD^A {Altimeter Domain} and SD^I {INS Domain}. By combining the valuation functions in SD^A and SD^I with respect to invariants from Equation 6.4 and Equation 6.5 in pilot's domain,

$$S_a = \neg\varphi_5 \wedge \varphi_6 \wedge \varphi_8 \Rightarrow \nexists V^P_{\sim i} \tag{6.18}$$

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

$$S_i = \neg\varphi_3 \wedge \neg\varphi_4 \wedge \neg\varphi_6 \wedge \varphi_7 \wedge \varphi_8 \Rightarrow \exists V^P_i \tag{6.19}$$

Using the invariants, the information received from INS domain is not faulty and the pilot can valuate for correctness of the position data in that domain.

By combining Equation 6.18 and Equation 6.19,

$$MSDND(ES) = \exists w \in W\colon\ [w \vdash \Box f(s_a, s_i)] \wedge [w \models (\nexists V^P_{\sim i} \wedge \exists V^P_i] \tag{6.20}$$

Hence, the system is *not Nondeducible secure* to the pilot and the controller as they can deduce that altimeter is presenting incorrect information.

The ProVerif code in APPENDIX C proves that this attack is *not Nondeducible secure* when INS is used as an alternate information flow path. As can be interpreted from "RESULT not attacker(Accept[]) is false", the altimeter process is compromised.

6.4. PITOT STATIC SYSTEM FAILURE

This section presents two different scenarios to check if the compromised pitot static system can be identified by the pilot.

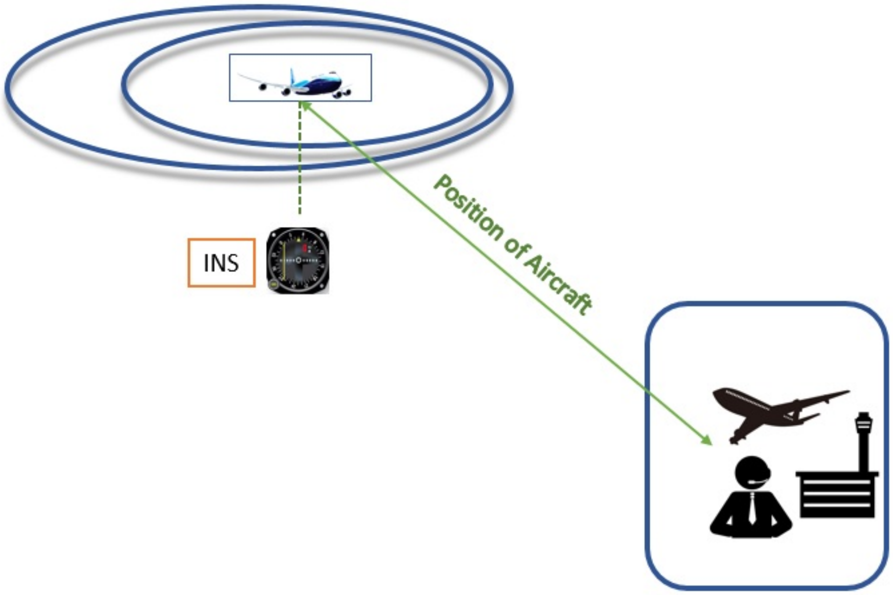

Figure 6.6. INS Helps To Identify The Faulty Altimeter

6.4.1. Scenario 1: Without Invariants. In case of a pitot static system compromise, the MSDND model yields Nondeducibility, thereby stopping critical information flow to the pilot.

Proof: Let us consider the possibility of failure in the altimeter of the pitot static system. Errors in pitot-static system altimeter readings can be extremely dangerous as the information obtained from the pitot static system, such as altitude, is potentially safety-critical. Several commercial airline disasters have been traced to a failure of the pitot-static system (See Figure 6.7).

Table 6.7 presents the set of logical conditions, φ_i, p, t, ps that can be evaluated to determine the interactions between the pilot and ATC controller.

Consider the scenario in which the air data computer presents a discrepancy in the readings of the air speed. Airspeed is probably the most important single piece of information the pilot needs. Virtually every phase of flight is conducted at a prescribed airspeed or range of airspeeds.

Table 6.7. Pitot Static System Logical Conditions And States - Without Invariants

φ_i	States	Functions
φ_0	Normal	Plane functions normally.
φ_1	Invariant Data	Initiation of transmission of pressure, speed, altitude and temperature from the sensors.
φ_2	Altimeter Data	Initiation of transmission of altitude reading to the TCAS system.
φ_3	Airspeed Data	Initiation of transmission of airspeed data to the pitot system.
φ_4	TCAS Data	TCAS System knows the position of the plane and the altitude data.
φ_5	Pitot Data	Pilot could see the air speed, altitude readings on the Pitot static system.
φ_6	Verification	Pilot checks for consistency of the data by verifying other systems using the available data.
φ_7	Failure	Abnormal Function.
S_p	p = T	$\boldsymbol{p = \neg\varphi_3 \wedge \varphi_4 \wedge \neg\varphi_5 \wedge \varphi_6 \wedge \varphi_7}$ ~df(Airspeed data) ∧ df(TCAS data) ∧ ~df(Pitot data) ∧ df(Verification) Output = TA/RA and Pitot readings
S_t	t = T	$\boldsymbol{t = \neg\varphi_1 \wedge \neg\varphi_2 \wedge \neg\varphi_4 \wedge \varphi_6 \wedge \varphi_7}$ ~df(Invariant Data) ∧ ~df(Altimeter Data) ∧ ~df(TCAS Data) ∧ df(Verification) Output = Altitude
S_{ps}	ps = T	$\boldsymbol{ps = \neg\varphi_1 \wedge \neg\varphi_3 \wedge \neg\varphi_5}$ ~df(Invariant Data) ∧ ~df(Airspeed data) ∧ ~df(Pitot data) Output = Pitot Data

The two security domains in this scenario are SD^T {TCAS domain} and SD^{PS} {pitot static system domain}. By combining the valuation functions in SD^{PS} and SD^T,

$$S_{ps} = \neg\varphi_1 \wedge \neg\varphi_3 \wedge \neg\varphi_5 \Rightarrow \nexists V^{PS}_{\sim p} \tag{6.21}$$

Since the information received from the pitot static system domain is faulty, the pilot cannot valuate for correctness of the altimeter data in that domain.

$$S_t = \neg\varphi_1 \wedge \neg\varphi_2 \wedge \neg\varphi_4 \Rightarrow \nexists V^{T}_{\sim p} \tag{6.22}$$

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

By combining Equation 6.21 and Equation 6.22,

$$MSDND(ES) = \exists w \in W : [w \vdash \Box f(S_{ps}, S_t)] \wedge [w \models (\nexists V^{PS}_{\sim p} \wedge \nexists V^{T}_{p}] \tag{6.23}$$

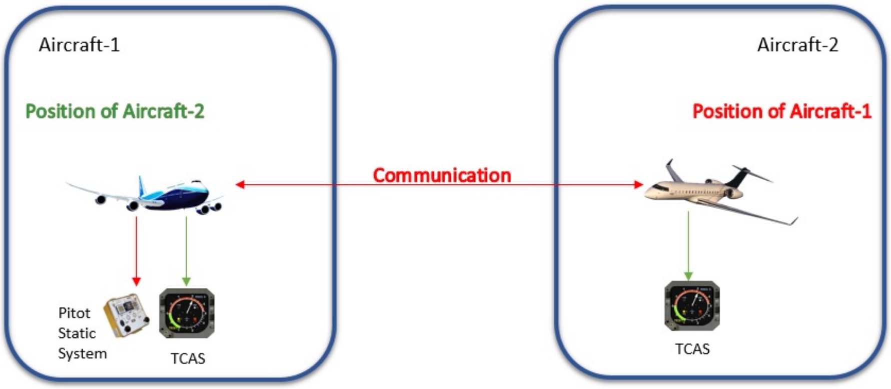

Figure 6.7. Pitot Static System Failure

Therefore, the pilot cannot deduce if the TCAS system is faulty or the pitot static system is faulty and identify the system that is compromised.

Hence, the system is *Nondeducible secure* to the pilot as he cannot deduce actual altimeter reading of the plane when there is a mismatch between the readings of TCAS and pitot static system.

6.4.2. Scenario 2: Using Invariants. By adding invariants, the MSDND model yields deducibility, thereby allowing critical information flow to the pilot when the pitot static system is compromised.

Proof: In order to make the system deducible to the pilot, invariants are used to express the correctness of the system. In case of failure in the pitot tube, there is a possibility to verify the correctness of the entire system by using other sources such as the Lift Reserve Indicator (LRI) or alternate static source (See Figure 6.8).

The lift of an aircraft is calculated using the formula below.

$$L = (1/2)dv^2sCL \tag{6.24}$$

where, d is the density of the air, v is velocity of aircraft, s is the area of wing, CL is the coefficient of lift.

In compressible subsonic flow the total pressure p_t is

$$P_t = P_s\left[1 + \frac{\gamma - 1}{\gamma}.\frac{\rho\delta^2}{2p_s}\right]^{\frac{\gamma-1}{\gamma}} \tag{6.25}$$

where, γ = specific heat ratio of air (= 1.4), V = true airspeed and ρ = free-stream air density.

From the above formula, true air speed can be calculated with the help of static pressure by the alternate static system and this ensures that the system is not compromised in such a case.

By considering the above possibilities, the correctness of the system can be verified in case of a failure in the pitot static system by relying on the data from alternate static source and LRI.

Table 6.8 presents the set of logical conditions, φ_i, p, i, ps, t that can be evaluated to determine the interactions between the pilot, TCAS and pitot static systems.

Table 6.8. Pitot Static System Logical Conditions And States - Using Invariants

φ_i	**States**	**Functions**
φ_0	Normal	Plane functions normally.
φ_1	Pitot Data	Initiation of transmission of pressure, speed, altitude and temperature from the pitot tube and static ports.
φ_2	Altimeter Data	Initiation of transmission of altitude reading to the TCAS system.
φ_3	Vertical Speed Indicator Data	Initiation of transmission of vertical speed to the pitot system.
φ_4	Airspeed Data	Initiation of transmission of airspeed data to the pitot system.
φ_5	Altitude Data	Initiation of transmission of altitude data to the TCAS system from XPDR.
φ_6	TCAS Altitude	TCAS System knows the position of the plane and the altitude data.
φ_7	L	Pilot knows correct airspeed value of the aircraft from the LRI system.
φ_8	δ	Pilot knows correct altitude of the aircraft from the alternate static source system.
φ_9	Verification	Pilot checks for consistency of the data by verifying invariant data from LRI and alternate static source.
S_p	p = T	$\boldsymbol{p = \neg\varphi_1 \wedge \neg\varphi_2 \wedge \neg\varphi_3 \wedge \neg\varphi_4 \wedge \varphi_6 \wedge \varphi_7 \wedge \neg\varphi_8 \wedge \varphi_9}$ ~df(Pitot Data) ∧ ~df(Altimeter Data) ∧ ~ df(Vertical Speed Indicator Data) ∧ ~ df(Airspeed Data) ∧ df(TCAS Altitude) ∧ df(L) ∧ df(V) ∧ df(Verification) Output = TA/RA and Pitot readings

S_i	i = T	$\boldsymbol{i = \varphi_7 \wedge \varphi_8 \wedge \varphi_9}$
		df(L) ∧ df(V) ∧ df(Verification)
		Output = Altitude

S_{ps}	ps = T	$\boldsymbol{ps = \neg\varphi_1 \wedge \neg\varphi_2 \wedge \neg\varphi_3 \wedge \neg\varphi_4 \wedge \neg\varphi_8}$ ~ df(Pitot Data) ∧ ~ df(Altimeter Data) ∧ ~ df(Vertical Speed Indicator Data) ∧ ~ df(Airspeed Data) Output = Pitot Data

S_t	t = T	$\boldsymbol{t = \varphi_5 \wedge \varphi_6}$
		df(Altitude Data) ∧ df(TCAS Altitude)
		Output = TA/RA

In order to verify the correctness of the airspeed data, two security domains SD^P {Pitot Static System} and SD^L {LRI System} are considered. By combining the valuation functions in SD^P and SD^L with respect to invariants from Equation 6.24,

$$S_{ps} = \neg\varphi_1 \wedge \neg\varphi_2 \wedge \neg\varphi_3 \wedge \neg\varphi_4 \Rightarrow \nexists V^P_{\sim i} \tag{6.26}$$

As before, the information received from the pitot static system domain is faulty and could not valuate for correctness of the airspeed data in that domain.

$$S_i = \varphi_5 \wedge \varphi_6 \wedge \varphi_7 \wedge \varphi_9 \Rightarrow \exists V^P_i \tag{6.27}$$

Using the invariants from Equation 6.24, solving for L, the information received from the LRI domain is not faulty and the correctness of airspeed can be valuated in that domain.

By combining Equation 6.26 and Equation 6.27,

$$MSDND(ES) = \exists w \in W\colon [w \vdash \Box f(s_{ps}, s_i)] \wedge [w \models (\nexists V^P_{\sim i} \wedge \exists V^P_i] \tag{6.28}$$

Therefore, the system is *not Nondeducible secure* to the pilot. The pilot can determine that the pitot static system is displaying incorrect airspeed data when there is a mismatch between the altitude reading of pitot static system and the TCAS system by using the data from LRI system.

In order to verify the correctness of altitude data, two security domains SD^{PS} {Pitot Static System} and SD^{AS} {Alternate Static Source System} are considered. By combining the valuation functions in SD^{PS} and SD^{AS} with respect to invariants from Equation 6.25,

$$S_{ps} = \neg\varphi_1 \wedge \neg\varphi_2 \wedge \neg\varphi_3 \wedge \neg\varphi_4 \Rightarrow \nexists V^P_{\sim i} \tag{6.29}$$

As before, the information received from the pitot static system domain is faulty and could not valuate for correctness of the altitude data in that domain.

$$S_{as} = \varphi_7 \wedge \varphi_8 \wedge \varphi_9 \Rightarrow \exists V_i^P \tag{6.30}$$

Using the invariants from Equation 6.25, solving for δ, the information received from the Alternate Static Source domain is not faulty and could valuate for correctness of the altitude data in that domain.

By combining Equation 6.29 and Equation 6.30,

$$MSDND(ES) = \exists w \in W : [w \vdash \Box f(s_{ps}, s_{as})] \wedge [w \models (\nexists V_{\sim i}^P \wedge \exists V_i^P] \tag{6.31}$$

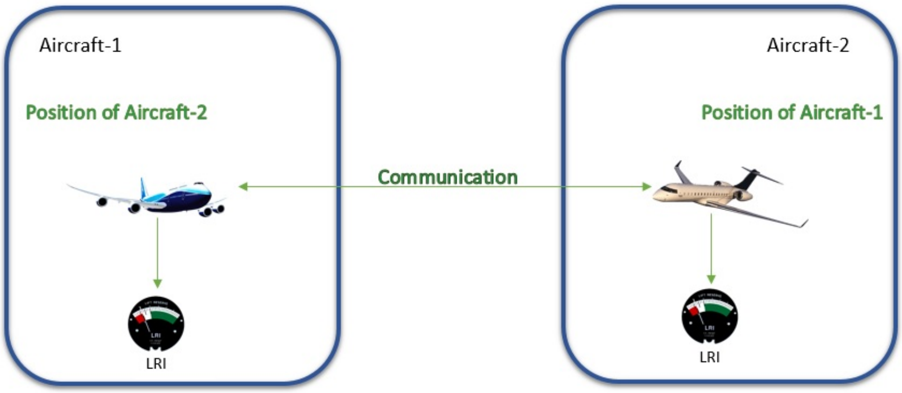

Figure 6.8. LRI Helps To Identify The Faulty Pitot Static System

Hence, the system is *not Nondeducible secure* to the pilot. The pilot can determine that the pitot static system is displaying incorrect data when there is a mismatch between the altitude reading of pitot static system and the TCAS system by using the data from alternate static source system.

Similarly, if there is a problem with the TCAS system, the pitot static system can be used as the reference and rely on the data from alternate static source to verify the correctness of the altitude displayed on the TCAS system.

The ProVerif code in APPENDIX D proves that this attack is *not Nondeducible secure* by using the alternate static source as an alternate information flow path to prove observational equivalence. As can be interpreted from "RESULT Observational equivalence is true (bad not derivable)", the attack can be deduced by pilot-1.

6.5. ADS-B TRANSMITTER FAILURE

This section presents two different scenarios to check if the compromised ADS-B transmitter can be identified by the pilot.

6.5.1. Without Invariants. In case of an ADS-B transmitter compromise, the MSDND model yields Nondeducibility, thereby stopping critical information flow to the pilot.

Proof: Consider a scenario in which two aircraft, Plane-1 and Plane-2 are 10 nautical miles (nmi) apart from each other. If there is a compromise in the ADS-B transmitter of the Plane-1, the position information of Plane-1 received by Plane-2 is incorrect and the same is displayed to the pilot in Plane-2. This could lead to improper communication between both the pilots. Hence, the pilots can deduce that something is going wrong but cannot distinguish which information is correct (See Figure 6.9).

Table 6.9 presents the set of logical conditions, $\varphi_i, p_1, p_2, a1, a2$ that can be evaluated to determine the interactions between pilot-1 and pilot-2.

Once the flight position is retrieved, pilots can observe that there is a mismatch between the ADS-B position data. The pilot-1 cannot distinguish whether there is a failure in the ADS-B-1 system or the ADS-B-2 system.

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},

$$S_{p1} = \neg\varphi_1 \wedge \varphi_3 \wedge \neg\varphi_5 \wedge \neg\varphi_7 \Rightarrow \nexists V^{P1}_{\sim a1} \tag{6.32}$$

Table 6.9. ADS-B Logical Conditions And States - Without Invariants

φ_i	**States**	**Functions**
φ_0	Normal	Plane functions normally.
φ_1	ADS-B Out$_1$	Initiation of transmission of position data from Plane 1.
φ_2	ADS-B Out$_2$	Initiation of transmission of position data from Plane 2.
φ_3	ADS-B In$_1$	Reception of position data from Plane 2.
φ_4	ADS-B In$_2$	Reception of position data from Plane 1.
φ_5	Display$_1$	Pilot-1 sees the position of the Plane-2.
φ_6	Display$_2$	Pilot-2 sees the position of the Plane-1.
φ_7	Communication	Correct communication between pilots from Plane-1 and Plane-2 based on the ADS-B data from both the aircraft.
S_{p1}	p1 = T	$\boldsymbol{p1 = \neg\varphi_1 \wedge \varphi_3 \wedge \neg\varphi_5 \wedge \neg\varphi_7}$ ~df(ADS-B Out$_1$)∧ df(ADS-B In$_1$)∧ ~df(Display$_1$)∧ ~df(communication) Output = *Position of Plane-2*

S_{p2}	p2 = T	$\boldsymbol{p2 = \varphi_2 \wedge \neg\varphi_4 \wedge \varphi_6 \wedge \neg\varphi_7}$
		df(ADS-B Out$_2$)∧ ~df(ADS-B In$_2$)∧ df(Display$_2$)∧ ~df(communication)
		Output = *Position of Plane-1*

S_{a1}	$a1$ = T	$\boldsymbol{a1 = \neg\varphi_1 \wedge \varphi_3 \wedge \neg\varphi_5}$ ~df(ADS-B Out$_1$)∧ df(ADS-B In$_1$)∧ ~df(Display$_1$) *Output = Position of Plane-2*
S_{a2}	$a2$ = T	$\boldsymbol{a2 = \varphi_2 \wedge \neg\varphi_4 \wedge \varphi_6}$ df(ADS-B Out$_2$)∧ ~df(ADS-B In$_2$)∧ df(Display$_2$) *Output = Position of Plane-1*

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

$$S_{p2} = \neg\varphi_4 \wedge \varphi_6 \wedge \neg\varphi_7 \Rightarrow \nexists V_{a1}^{P2} \tag{6.33}$$

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

By combining Equation 6.32 and Equation 6.33,

$$MSDND(ES) = \exists w \in W : [w \vdash \Box f(S_{p1}, S_{p2})] \wedge [w \models (\nexists V^{P1}_{\sim a1} \wedge \nexists V^{P2}_{a1}] \tag{6.34}$$

Therefore, the pilots from both the aircraft cannot deduce the faulty ADS-B system and identify the system that is compromised.

Hence, the system is *Nondeducible secure* to the pilots as they cannot deduce the actual true position of plane-2.

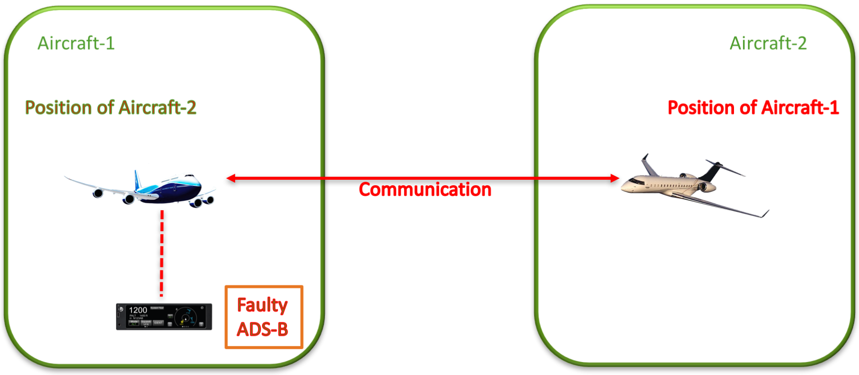

Figure 6.9. Failure In ADS-B Transmitter

6.5.2. Using Invariants. In case of ADS-B transmitter compromise, but using INS, the MSDND model yields deducibility thereby allowing critical information flow to the pilot.

Proof: The above mentioned scenario can be made deducible to the pilot by relying on an alternate system; i.e., INS which helps in figuring out the compromised system. Hence, the pilots can deduce the source of incorrect information based on the invariant data computed by the INS (See Figure 6.10).

In case of failure in the ADS-B transmitter, the pilot can make use of the INS system. The velocity is calculated using Equation 6.4.

Using the final velocity, distance travelled by the aircraft can be calculated using Equation 6.5.

Note: In Table 6.10, d_1 is the distance of plane-1 projected on INS-1 system and d_2 is the distance travelled by plane-2 projected on INS-2 system over the same time interval.

Table 6.10 presents the set of logical conditions, $\varphi_i, p_1, p_2, a1, a2, d_1, d_2, i_1, i_2$ that can be evaluated to determine the interactions between the pilot-1 and pilot-2.

The two security domains in this scenario are SD^A {ADS-B Domain} and SD^I {INS Domain}. By combining the valuation functions in SD^A and SD^I with respect to invariants from Equation 6.4 and Equation 6.5 in pilot's domain,

$$S_{a1} \wedge S_{a2} = \neg\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \neg\varphi_4 \wedge \neg\varphi_5 \wedge \varphi_6 \Rightarrow \nexists V_{\sim i}^{P1} \tag{6.35}$$

Since the information received from the ADS-B domain is faulty, the pilot cannot valuate for correctness of the position data in that domain.

$$S_{i1} \wedge S_{i2} = \varphi_8 \wedge \varphi_9 \wedge \varphi_{10} \Rightarrow \exists V_i^{P1} \tag{6.36}$$

Using the invariants, the information received from INS domain is not faulty and the pilot can valuate for correctness of the position data in that domain.

By combining Equation 6.35 and Equation 6.36,

$$MSDND(ES) = \exists w \in W\colon [w \vdash \Box f(s_{a1}, s_{i1})] \wedge [w \models (\nexists V_{\sim i}^{P1} \wedge \exists V_i^{P1}] \tag{6.37}$$

Hence, the system is *not Nondeducible secure* to pilot-1 and he can deduce that ADS-B is presenting incorrect information.

Table 6.10. ADS-B Logical Conditions And States - Using Invariants

φ_i	**States**	**Functions**
φ_0	Normal	Plane functions normally.
φ_1	ADS-B Out$_1$	Initiation of transmission of position data from Plane 1.
φ_2	ADS-B Out$_2$	Initiation of transmission of position data from Plane 2.
φ_3	ADS-B In$_1$	Reception of position data from Plane 2.
φ_4	ADS-B In$_2$	Reception of position data from Plane 1.
φ_5	Display$_1$	Pilot-1 sees the position of the Plane-2.
φ_6	Display$_2$	Pilot-2 sees the position of the Plane-1.
φ_7	Communication	Correct communication between pilots from Plane-1 and Plane-2 based on the ADS-B data from both the aircraft.
φ_8	d_1	Pilot-1 checks the position of the plane-1 and verifies it with the ADS-B data.
φ_9	d_2	Pilot-2 checks the position of the plane-2 and verifies it with the ADS-B data.
φ_{10}	Verification	Pilot-1 and Pilot-2 checks and verifies the position of the respective aircraft and verifies it with the position data displayed.
S_{p1}	p1 = T	$\boldsymbol{p1 = \neg\varphi_1 \wedge \varphi_3 \wedge \neg\varphi_5 \wedge \neg\varphi_7 \wedge \varphi_8 \wedge \varphi_{10}}$ ~df(ADS-B Out$_1$)$\wedge$ df(ADS-B In$_1$)$\wedge$ ~df(Display$_1$)$\wedge$ ~df(communication) $\wedge$ df(d_1) $\wedge$ df(Verification) Output = *Position of Plane-2*

S_{p2}	p2 = T	$\boldsymbol{p2 = \varphi_2 \wedge \neg\varphi_4 \wedge \varphi_6 \wedge \neg\varphi_7 \wedge \varphi_8 \wedge \varphi_9 \wedge \varphi_{10}}$
		df(ADS-B Out$_2$)$\wedge$ ~df(ADS-B In$_2$)$\wedge$ df(Display$_2$)$\wedge$ ~df(communication) $\wedge$ df(d_2) $\wedge$ df(Verification)
		Output = *Position of Plane-1*

S_{a1}	$a1$ = T	$\boldsymbol{a1 = \neg\varphi_1 \wedge \varphi_3 \wedge \neg\varphi_5}$ ~df(ADS-B Out$_1$)$\wedge$ df(ADS-B In$_1$)$\wedge$ ~df(Display$_1$) *Output = Position of Plane-2*
S_{a2}	$a2$ = T	$\boldsymbol{a2 = \varphi_2 \wedge \neg\varphi_4 \wedge \varphi_6}$ df(ADS-B Out$_2$)$\wedge$ ~df(ADS-B In$_2$)$\wedge$ df(Display$_2$) *Output = Position of Plane-1*

S_{i1}	$i1$ = T	$\boldsymbol{i1 = \varphi_8 \wedge \varphi_{10}}$
		df(d_1) $\wedge$ df(Verification)
		Output = *Position of Plane-1*

S_{i2}	$i2$ = T	$\boldsymbol{i2 = \varphi_9 \wedge \varphi_{10}}$
		df(d_2) $\wedge$ df(Verification)
		Output = *Position of Plane-2*

The ProVerif code in APPENDIX E proves that this attack is *not Nondeducible secure* by using INS as an alternate information flow to prove observational equivalence. As can be interpreted from "RESULT Observational equivalence is true (bad not derivable)", the attack can be deduced by the pilots.

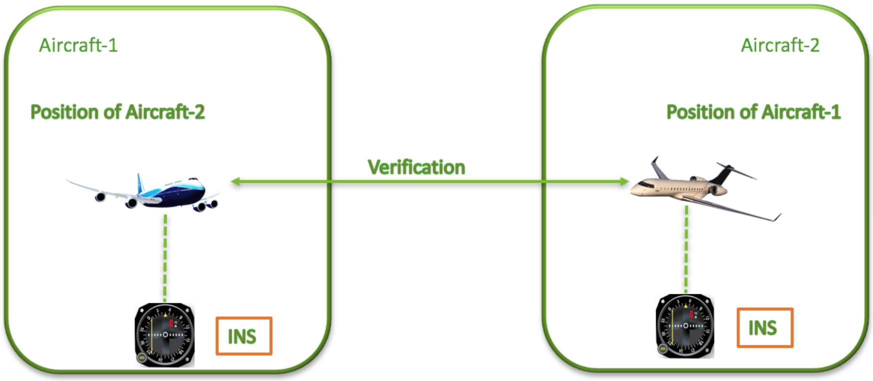

Figure 6.10. INS Helps To Identify The Failure In ADS-B transmitter

6.6. ADS-B, INS AND ATTACKER

This section presents two different scenarios to check if the attacker plane can be identified by the pilot using INS and ADS-B systems.

6.6.1. Scenario 1: Without Using Invariants. In case of an attacker sending fake signals, the MSDND model yields Nondeducibility, thereby stopping critical information flow to the pilot.

Proof: Consider a scenario in which two aircraft, Plane-1 and Plane-2 are 10nmi apart from each other. If pilot-2 sends fake position data to Plane-1, the position information received by Plane-1 is incorrect. This could lead to improper communication between both the pilots. Hence, pilot-1 can deduce that something is going wrong from the ADS-B data and INS data, but cannot distinguish which information is correct (See Figure 6.11).

Table 6.11 presents the set of logical conditions, $\varphi_i, p_1, p_2, a1, a2, i1, i2$ that can be evaluated to determine the interactions between pilot-1 and pilot-2.

Table 6.11. ADS-B And Attacker Logical Conditions And States - Without Invariants

φ_j	States	Functions
φ_0	Normal	Plane functions normally.
φ_1	ADS-B Out$_1$	Initiation of transmission of position data from Plane 1.
φ_2	ADS-B Out$_2$	Initiation of transmission of position data from Plane 2.
φ_3	ADS-B In$_1$	Reception of position data from Plane 2.
φ_4	ADS-B In$_2$	Reception of position data from Plane 1.
φ_5	Display$_1$	Pilot-1 sees the position of the Plane-2.
φ_6	Display$_2$	Pilot-2 sees the position of the Plane-1.
φ_7	d$_1$	Pilot-1 checks the INS position of the plane-1 and verifies it with the ADS-B data.
φ_8	d$_2$	Pilot-2 checks the INS position of the plane-2 and verifies it with the ADS-B data.
φ_7	Verification	Pilot-1 and Pilot-2 checks and verifies the position of the respective aircraft and verifies it with the position data displayed.
S_{p1}	p1 = T	$\boldsymbol{p1 = \neg\varphi_1 \land \varphi_3 \land \neg\varphi_5 \land \neg\varphi_7}$ ~df(ADS-B Out$_1$)$\land$ df(ADS-B In$_1$)$\land$ ~df(Display$_1$)$\land$ ~df(communication) Output = *Position of Plane-2*

S_{p2}	p2 = T	$\boldsymbol{p2 = \varphi_2 \land \neg\varphi_4 \land \varphi_6 \land \neg\varphi_7}$
		df(ADS-B Out$_2$)$\land$ ~df(ADS-B In$_2$)$\land$ df(Display$_2$)$\land$ ~df(communication)
		Output = *Position of Plane-1*

S_{a1}	$a1$ = T	$\boldsymbol{a1 = \neg\varphi_1 \land \varphi_3 \land \neg\varphi_5}$ ~df(ADS-B Out$_1$)$\land$ df(ADS-B In$_1$)$\land$ ~df(Display$_1$) *Output = Position of Plane-2*
S_{a2}	$a2$ = T	$\boldsymbol{a2 = \varphi_2 \land \neg\varphi_4 \land \varphi_6}$ df(ADS-B Out$_2$)$\land$ ~df(ADS-B In$_2$)$\land$ df(Display$_2$) *Output = Position of Plane-1*

S_{i1}	$i1$ = T	$\boldsymbol{i1 = \varphi_8 \land \varphi_{10}}$
		df(d_1) $\land$ df(Verification)
		Output = *Position of Plane-1*

S$_{i2}$	i2 = T	$\boldsymbol{i2 = \varphi_9 \land \varphi_{10}}$ ~df(d_2) $\land$ df(Verification) Output = *Position of Plane-2*

Once the flight position is retrieved, pilot-1 trusts the information sent by Plane-2. Pilot-1 observes that there is a mismatch between the ADS-B position data and the INS-data.

Pilot-1 cannot distinguish whether there is a failure in the ADS-B-1 system or the INS system.

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},

$$S_{p1} = \neg\varphi_1 \wedge \varphi_3 \wedge \neg\varphi_5 \wedge \neg\varphi_7 \Rightarrow \nexists V^{P1}_{\sim a1} \tag{6.38}$$

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

$$S_{p2} = \neg\varphi_4 \wedge \varphi_6 \wedge \neg\varphi_7 \Rightarrow \nexists V^{P2}_{a1} \tag{6.39}$$

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

By combining Equation 6.38 and Equation 6.39,

$$MSDND(ES) = \exists w \in W\colon [w \vdash \Box f(S_{p1}, S_{p2})] \wedge [w \models (\nexists V^{P1}_{\sim a1} \wedge \nexists V^{P2}_{a1}] \tag{6.40}$$

Therefore, pilot-1 cannot deduce that pilot-2 is sending fake signals and identify the system that is compromised.

Hence, the system is *Nondeducible secure* to pilot-1 as he/she cannot deduce the actual true position of plane-2.

6.6.2. Scenario 2: Using Invariants. In case of an attacker sending fake signals, but using TCAS, the MSDND model yields deducibility thereby allowing critical information flow to the pilot.

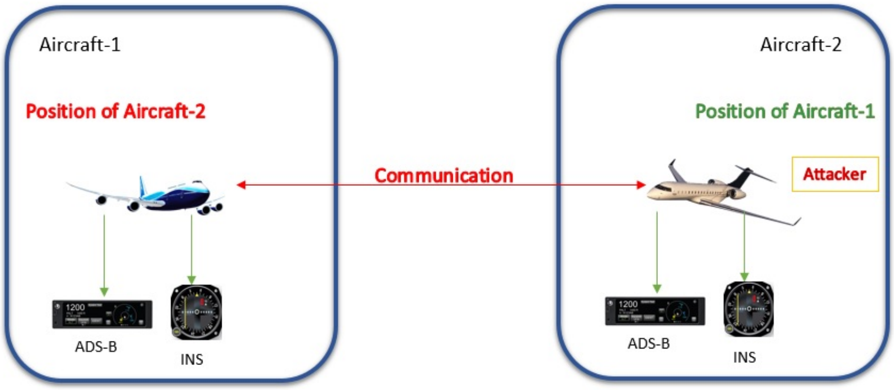

Figure 6.11. Attacker Sends Fake Position Data To Pilot-1

Proof: The above mentioned scenario can be made deducible to pilot-1 by relying on an alternate system; i.e., TCAS which helps in figuring out the compromised system. Hence, pilot-1 can deduce the source of incorrect information based on the invariant data computed by TCAS.

Consider a scenario in which two aircraft, Plane-1 and Plane-2 are 10nmi apart from each other. If pilot-2 sends fake position data to Plane-1, the position information received by Plane-1 is incorrect. This could lead to improper communication between both the pilots. Hence, pilot-1 can deduce that something is going wrong from the ADS-B data and INS data, but cannot distinguish which information is correct (See Figure 6.12).

Table 6.12 presents the set of logical conditions, $\varphi_i, p_1, p_2, a1, a2, t$ that can be evaluated to determine the interactions between pilot-1 and pilot-2.

Once the flight position is retrieved from TCAS-1, pilot-1 does not trust the information sent by Plane-2. Pilot-1 relies on the TCAS data and follows the RA given by TCAS-1 system.

TCAS computes the closure rate of each target within surveillance range based on the surveillance reports (slant range, bearing and altitude) provided each second, in order to determine the time in seconds to Closest Point of Approach (CPA), and the horizontal miss distance at CPA. If the target aircraft is equipped with an altitude-coding transponder, the

Table 6.12. ADS-B And Attacker Logical Conditions And States - Using Invariants

φ_i	States	Functions
φ_0	Normal	Plane functions normally.
φ_1	ADS-B Out$_1$	Initiation of transmission of position data from Plane 1.
φ_2	ADS-B Out$_2$	Initiation of transmission of position data from Plane 2.
φ_3	ADS-B In$_1$	Reception of position data from Plane 2.
φ_4	ADS-B In$_2$	Reception of position data from Plane 1.
φ_5	Display$_1$	Pilot-1 sees the position of the Plane-2.
φ_6	Display$_2$	Pilot-2 sees the position of the Plane-1.
φ_7	TCAS-1	TCAS system knows the position of plane-1 and plane-2.
φ_8	Display$_t$	Pilot-1 checks the TCAS position of the plane-2 and follows the RA given by TCAS.
S_{p1}	p1 = T	**$p1 = \neg\varphi_1 \wedge \varphi_3 \wedge \neg\varphi_5 \wedge \varphi_7 \wedge \varphi_8$** ~df(ADS-B Out$_1$)∧ df(ADS-B In$_1$)∧ ~df(Display$_1$)∧ ~df(communication) Output = *Position of Plane-2*

S_{p2}	p2 = T	**$p2 = \varphi_2 \wedge \neg\varphi_4 \wedge \varphi_6 \wedge \neg\varphi_7$**
		df(ADS-B Out$_2$)∧ ~df(ADS-B In$_2$)∧ df(Display$_2$)∧ ~df(communication)
		Output = *Position of Plane-1*

S_{a1}	$a1$ = T	**$a1 = \neg\varphi_1 \wedge \varphi_3 \wedge \neg\varphi_5$** ~df(ADS-B Out$_1$)∧ df(ADS-B In$_1$)∧ ~df(Display$_1$) *Output = Position of Plane-2*
S_{a2}	$a2$ = T	**$a2 = \varphi_2 \wedge \neg\varphi_4 \wedge \varphi_6$** df(ADS-B Out$_2$)∧ ~df(ADS-B In$_2$)∧ df(Display$_2$) *Output = Position of Plane-1*
S_i	i = T	**$t = \varphi_7 \wedge \varphi_8$** df(TCAS-1) ∧ df(Display$_t$) *Output = Position of Plane-1 and Plane-2*

TCAS calculates the altitude of the target at CPA. The intruder's vertical speed is obtained by measuring the time it takes to cross successive 100-foot or 25- foot altitude increments, which depends upon the type of altitude coding transponder. The TCAS system uses the data from its own aircraft pressure altimeter, either directly from the altitude encoder or Air Data Computer (ADC). In this way, it determines its own aircraft altitude, vertical rate, and the relative altitude of each target. The outputs from the TCAS tracking algorithm (target range, horizontal miss distance at CPA, closure rate and relative altitude of the target aircraft) are supplied to the traffic advisory and threat detection algorithms.

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} with respect to invariants in them from TCAS domain,

$$S_{p1} = \neg\varphi_1 \wedge \varphi_3 \wedge \neg\varphi_5 \wedge \varphi_7 \wedge \varphi_8 \Rightarrow \exists V_i^{P1} \tag{6.41}$$

Since the information received from the TCAS-1 domain is not faulty, pilot-1 can valuate for correctness of the position data in that domain.

$$S_{p2} = \neg\varphi_4 \wedge \varphi_6 \wedge \neg\varphi_7 \Rightarrow \nexists V_{\sim i}^{P1} \tag{6.42}$$

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

By combining Equation 6.41 and Equation 6.42,

$$MSDND(ES) = \exists w \in W\colon\ [w \vdash \Box f(S_{p1}, S_{p2})] \wedge [w \models (\exists V_i^{P1} \wedge \nexists V_i^{P1}] \tag{6.43}$$

Therefore, pilot-1 can deduce that pilot-2 is sending fake position data and identify the system that is compromised.

Hence, the system is *not Nondeducible secure* to pilot-1 as he cannot deduce the actual true position of plane-2.

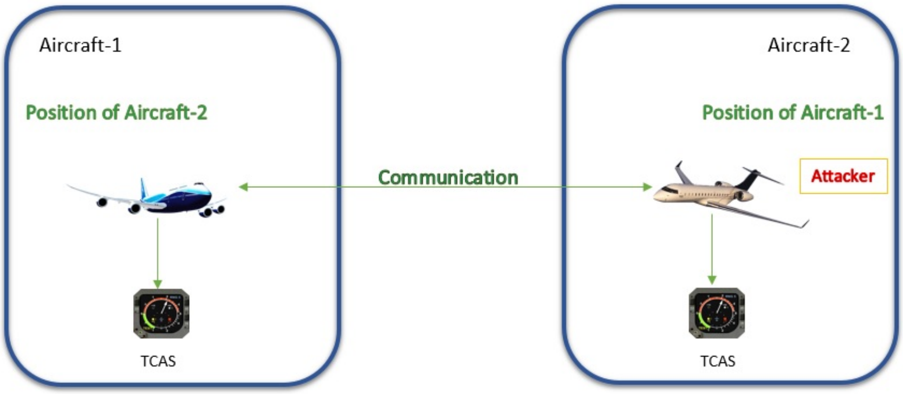

Figure 6.12. TCAS Helps To Identify The Attacker Plane

The ProVerif code in APPENDIX F proves that this attack is *not Nondeducible secure* when the TCAS system is used as an alternate information flow path. As can be interpreted from "RESULT not attacker(XY_Coord_Node_1[]) is false", the attacker plane is present in coordinate 1.

6.7. ADS-B AND RF INTERFERENCE

In case of RF interference, the MSDND model yields Nondeducibility, thereby stopping critical information flow to the pilots.

Proof: Consider a scenario in which two aircraft, Plane-1 and Plane-2 are 10nmi apart from each other. The pilots communicate with each other on 1090ES band. In case of an RF interference, the communication path between both the planes is disrupted and incorrect information is transmitted to both the pilots. Hence, the pilots cannot deduce that something is going wrong and this could lead to a potential collision (See Figure 6.13).

Table 6.13 presents the set of logical conditions, $\varphi_i, p_1, p_2, a1, a2$ that can be evaluated to determine the interactions between pilot-1 and pilot-2.

Table 6.13. RF Interference Logical Conditions And States - Without Invariants

φ_j	States	Functions
φ_0	Normal	Plane functions normally.
φ_1	ADS-B Out$_1$	Initiation of transmission of position data from Plane 1.
φ_2	ADS-B Out$_2$	Initiation of transmission of position data from Plane 2.
φ_3	RF Interference	RF Interference caused by the attacker.
φ_4	ADS-B In$_1$	Reception of position data from Plane 2.
φ_5	ADS-B In$_2$	Reception of position data from Plane 1.
φ_6	Display$_1$	Pilot-1 sees the position of the Plane-2.
φ_7	Display$_2$	Pilot-2 sees the position of the Plane-1.
φ_8	Communication	Incorrect communication between pilots from Plane-1 and Plane-2 based on the received ADS-B data.

S$_{p1}$	p1 = T	$\boldsymbol{p1 = \varphi_1 \wedge \neg\varphi_3 \wedge \neg\varphi_4 \wedge \neg\varphi_6 \wedge \neg\varphi_8}$
		df(ADS-B Out$_1$)∧ ~df(RF Interference) ∧ ~df(ADS-B In$_1$)∧ ~df(Display$_1$)∧ ~df(communication)
		Output = *Position of Plane-2*

S_{p2}	p2 = T	$\boldsymbol{p2 = \varphi_2 \wedge \neg\varphi_3 \wedge \neg\varphi_5 \wedge \neg\varphi_7 \wedge \neg\varphi_8}$
		df(ADS-B Out$_2$)∧ ~df(RF Interference) ∧ ~df(ADS-B In$_2$)∧ ~df(Display$_2$)∧ ~df(communication)
		Output = *Position of Plane-1*

S_{a1}	$a1$ = T	$\boldsymbol{a1 = \neg\varphi_1 \wedge \varphi_3 \wedge \neg\varphi_5}$ ~df(ADS-B Out$_1$)∧ df(ADS-B In$_1$)∧ ~df(Display$_1$) *Output = Position of Plane-2*
S_{a2}	$a2$ = T	$\boldsymbol{a2 = \varphi_2 \wedge \neg\varphi_4 \wedge \varphi_6}$ df(ADS-B Out$_2$)∧ ~df(ADS-B In$_2$)∧ df(Display$_2$) *Output = Position of Plane-1*

The two security domains in this scenario are SD^A {ADS-B Domain} and SD^I {INS Domain}. By combining the valuation functions in SD^A and SD^I with respect to invariants in them from pilot's domain,

$$S_{a1} \wedge S_{a2} = \neg\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \neg\varphi_4 \wedge \neg\varphi_5 \wedge \varphi_6 \Rightarrow \nexists V_{\sim i}^{P1} \tag{6.44}$$

Since the information received from the ADS-B domain is faulty, the pilot cannot valuate for correctness of the position data in that domain.

$$S_{i1} \wedge S_{i2} = \varphi_8 \wedge \varphi_9 \wedge \varphi_{10} \Rightarrow \exists V_i^{P1} \tag{6.45}$$

By combining Equation 6.44 and Equation 6.45,

$$MSDND(ES) = \exists w \in W : [w \vdash \Box f(s_{a1}, s_{i1})] \wedge [w \models (\nexists V_{\sim i}^{P1} \wedge \exists V_i^{P1}] \tag{6.46}$$

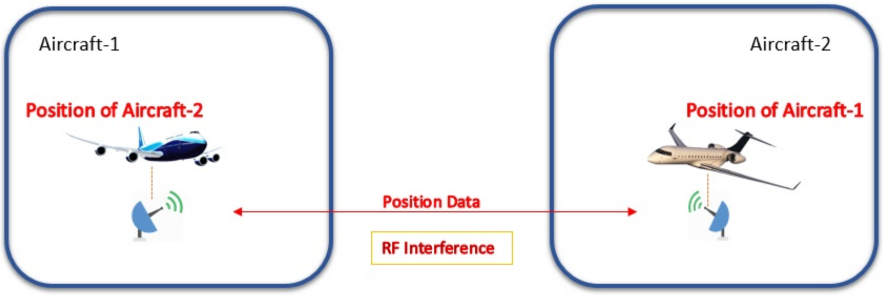

Figure 6.13. RF Interference

Hence, the system is *not Nondeducible secure* to pilot-1 and he/she can deduce that ADS-B is presenting incorrect information.

The ProVerif code in APPENDIX G proves that this attack is *Nondeducible secure* as there is no alternate information flow to prove observational equivalence. As can be interpreted from "RESULT Observational equivalence cannot be proved (bad derivable)", the attack cannot be deduced by the pilots.

6.8. ADS-B AND SATELLITE FAILURE

In case of satellite (GNSS) failure, the MSDND model yields Nondeducibility, thereby stopping critical information flow to the pilots.

Proof: Consider a scenario in which two aircraft, Plane-1 and Plane-2 are 10nmi apart from each other. GNSS is responsible for sending the position data to the planes and the planes communicate with each other. In case of GNSS failure, the position information retrieved by both the planes is incorrect and the pilots communicate with each other based on this information. Hence, the pilots cannot deduce that something is going wrong and this could lead to a potential collision (See Figure 6.14).

Table 6.14 presents the set of logical conditions, $\varphi_i, p_1, p_2, a1, a2$ that can be evaluated to determine the interactions between pilot-1 and pilot-2.

Table 6.14. Satellite Failure Logical Conditions And States - Without Invariants

φ_i	States	Functions
φ_0	Normal	Plane functions normally.
φ_1	GNSS	Initiation of transmission of position data to plane-1 and plane-2.
φ_2	ADS-B Out$_1$	Initiation of transmission of position data from Plane-1.
φ_3	ADS-B Out$_2$	Initiation of transmission of position data from Plane-2.
φ_4	ADS-B In$_1$	Reception of position data from Plane 2.
φ_5	ADS-B In$_2$	Reception of position data from Plane 1.
φ_6	Display$_1$	Pilot-1 sees the position of the Plane-2.
φ_7	Display$_2$	Pilot-2 sees the position of the Plane-1.
φ_8	Communication	Incorrect communication between pilots from Plane-1 and Plane-2 based on the received ADS-B data.
S$_{p1}$	p1 = T	$\boldsymbol{p1 = \neg\varphi_1 \wedge \neg\varphi_2 \wedge \neg\varphi_4 \wedge \neg\varphi_6 \wedge \neg\varphi_8}$ ~df(GNSS) ∧ ~df(ADS-B Out$_1$)∧ ~df(ADS-B In$_1$)∧ ~df(Display$_1$)∧ ~df(communication) Output = *Position of Plane-2*
S_{p2}	p2 = T	$\boldsymbol{p2 = \neg\varphi_1 \wedge \neg\varphi_3 \wedge \neg\varphi_5 \wedge \neg\varphi_7 \wedge \neg\varphi_8}$ ~df(GNSS) ∧ ~df(ADS-B Out$_2$)∧ ~df(ADS-B In$_2$)∧ ~df(Display$_2$)∧ ~df(communication) Output = *Position of Plane-1*
S_{a1}	$a1$ = T	$\boldsymbol{a1 = \neg\varphi_2 \wedge \neg\varphi_4 \wedge \neg\varphi_6}$ ~df(ADS-B Out$_1$)∧ ~df(ADS-B In$_1$)∧ ~df(Display$_1$) *Output = Position of Plane-2*
S_{a2}	$a2$ = T	$\boldsymbol{a2 = \neg\varphi_3 \wedge \neg\varphi_5 \wedge \neg\varphi_7}$ ~df(ADS-B Out$_2$)∧ ~df(ADS-B In$_2$)∧ ~df(Display$_2$) *Output = Position of Plane-1*

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 eventually leads to breakdown in the separation.

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},

$$S_{p1} = \neg\varphi_1 \wedge \neg\varphi_2 \wedge \neg\varphi_4 \wedge \neg\varphi_6 \wedge \neg\varphi_8 \Rightarrow \nexists V^{P1}_{\sim a2} \tag{6.47}$$

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

$$S_{p2} = \neg\varphi_1 \wedge \neg\varphi_3 \wedge \neg\varphi_5 \wedge \neg\varphi_7 \wedge \neg\varphi_8 \Rightarrow \nexists V^{P2}_{\sim a1} \tag{6.48}$$

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

By combining Equation 6.47 and Equation 6.48,

$$MSDND(ES) = \exists w \in W\colon [w \vdash \Box f(S_{p1}, S_{p2})] \wedge [w \models (\nexists V^{P1}_{\sim a2} \wedge \nexists V^{P2}_{\sim a1}] \tag{6.49}$$

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

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

The ProVerif code in APPENDIX H proves that this attack is *Nondeducible secure* as there is no alternate information flow to prove observational equivalence. As can be interpreted from "RESULT Observational equivalence cannot be proved (bad derivable)", the attack cannot be deduced by the pilots.

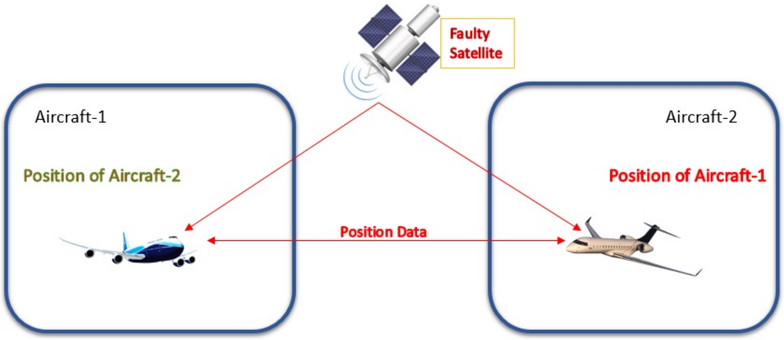

Figure 6.14. Satellite Failure

6.9. TCAS AND ATC CONTROLLER

This section presents two different scenarios to check if the incorrect commands from ATC can be identified by the pilot.

6.9.1. Scenario 1: Without Using Invariants. In case of incorrect commands from the ATC controller, the MSDND model yields Nondeducibility, thereby stopping critical information flow to the pilots.

Proof: Consider a scenario in which two aircraft, Plane-1 and Plane-2 are 10nmi apart from each other. The TCAS system sends climb RA to plane-1 and descend RA to plane-2. Due to delay in the reception of data from the both the planes, ATC instructs plane-2 to descend which induces pilot-2 to manoeuvre, overriding the initial RAs. Notably, reversing the on going RA is not feasible while aircraft are manoeuvring in the vertical dimension and are at co-altitude. This can lead to delaying the decision to reverse if both aircraft are climbing or descending at similar vertical speeds which eventually leads to collision (See Figure 6.15).

Table 6.15 presents the set of logical conditions, φ_i, p_1, p_2, c that can be evaluated to determine the interactions between pilot-1 and pilot-2.

Table 6.15. TCAS And ATC Logical Conditions And States - Without Invariants

φ_j	States	Functions
φ_0	Normal	Plane functions normally.
φ_1	TCAS-1	Initiation of transmission of "descend RA" to plane-1.
φ_2	TCAS-2	Initiation of transmission of "climb RA" to Plane-2.
φ_3	Display$_1$	Pilot-1 sees the position of the Plane-2 .
φ_4	Display$_2$	Pilot-2 sees the position of the Plane-1.
φ_5	ATC Command	Initiation of transmission of "descend RA" to Plane-2.
φ_6	Pilot-2	Pilot-2 follows the ATC instruction and descends.
φ_7	Communication	Incorrect communication between pilots from Plane-1 and Plane-2 based on the received RA's.

S$_{p1}$	p1 = T	$\boldsymbol{p1 = \varphi_1 \wedge \varphi_3 \wedge \neg\varphi_7}$
		df(TCAS-1) ∧ df(Display$_1$)∧ ~df(communication)
		Output = *Position of Plane-2*

S_{p2}	p2 = T	$\boldsymbol{p2 = \varphi_2 \wedge \varphi_4 \wedge \neg\varphi_5 \wedge \neg\varphi_6 \wedge \neg\varphi_7}$
		df(TCAS-2) ∧ df(Display$_2$)∧ ~df(ATC Command) ∧ ~df(Pilot-2) ∧ ~df(communication)
		Output = *Position of Plane-1*

S_c	*c* = T	$\boldsymbol{c = \varphi_1 \wedge \varphi_2 \wedge \neg\varphi_5 \wedge \neg\varphi_7}$
		df(TCAS-1) ∧ df(TCAS-2) ∧ ~df(ATC Command) ∧ ~df(Communication)
		Output = *Position of Plane-1 and Plane-2*

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

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},

$$S_{p1} = \varphi_1 \wedge \varphi_3 \wedge \neg\varphi_7 \Rightarrow \nexists V_c^{P1} \tag{6.50}$$

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

$$S_{p2} = \varphi_2 \wedge \varphi_4 \wedge \neg\varphi_5 \wedge \neg\varphi_6 \wedge \neg\varphi_7 \Rightarrow \nexists V_{\sim c}^{P2} \tag{6.51}$$

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

By combining Equation 6.50 and Equation 6.51,

$$MSDND(ES) = \exists w \in W\colon [w \vdash \Box f(S_{p1}, S_{p2})] \wedge [w \models (\nexists V_c^{P1} \wedge \nexists V_{\sim c}^{P2}] \tag{6.52}$$

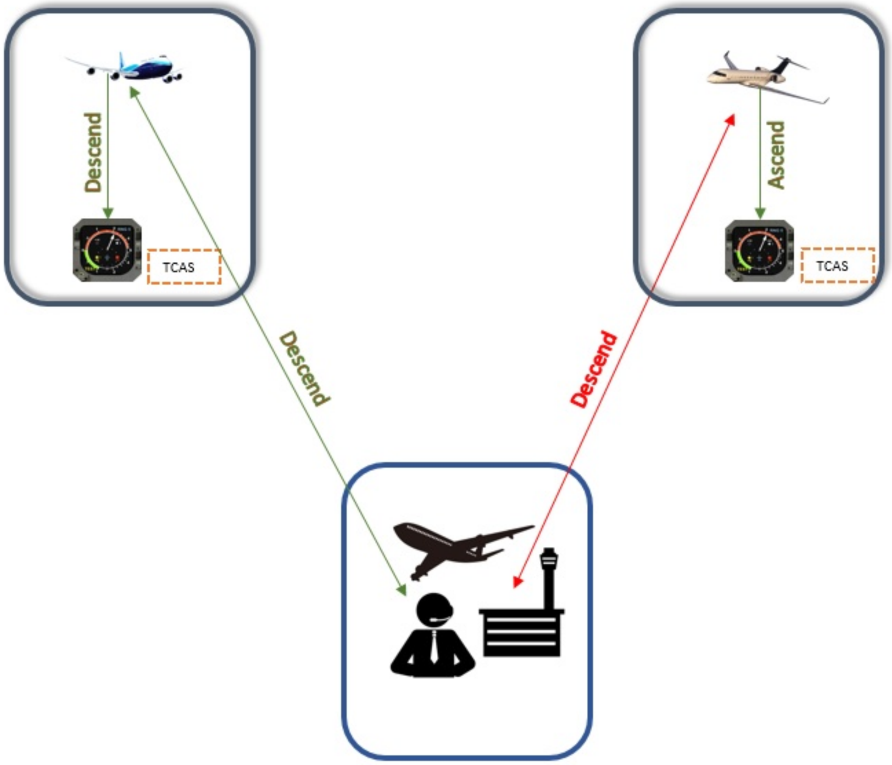

Figure 6.15. Conflict Between ATC And TCAS Commands

Therefore, the pilots from both the aircraft cannot deduce whether TCAS-2 is faulty or the ATC controller instructed wrong commands.

Hence, the system is *Nondeducible secure* to the pilots as they cannot deduce the correctness of ATC commands.

6.9.2. Scenario 2: Using Invariants. In case of incorrect commands from the ATC controller, but using ADS-B system, the MSDND model yields Nondeducibility, thereby allowing critical information flow to the pilots.

Proof: The above mentioned scenario can be avoided by relying on alternate information flow path. ADS-B system can serve as this alternate source to provide position data. This helps the pilots to verify the position when they are on the same flight level and in close proximity (See Figure 6.16).

Table 6.16 presents the set of logical conditions, $\varphi_i, p_1, p_2, c, a1, a2$ that can be evaluated to determine the interactions between pilot-1 and pilot-2.

Once the flight position is retrieved, pilot-1 trusts the information sent by Plane-2 and ATC controller and vice-versa. When both the pilots verify the TCAS position data and ADS-B position data, pilot-1 can deduce that ATC has given incorrect commands. Thus, the pilots can avoid collision by relying on the ADS-B data.

For each aircraft being tracked, the receiver calculates how long the satellite signal took to reach it, as follows:

Propagation Time = Time Signal Reached Receiver - Time Signal Left Satellite

$$P_t = R_t - L_t \tag{6.53}$$

Multiplying this propagation time by the speed of light gives the distance to the satellite.

$$S_d = P_t * S_l \tag{6.54}$$

Table 6.16. TCAS And ATC Logical Conditions And States - Using Invariants

φ_j	States	Functions
φ_0	Normal	Plane functions normally.
φ_1	TCAS-1	Initiation of transmission of "descend RA" to plane-1.
φ_2	TCAS-2	Initiation of transmission of "climb RA" to Plane-2.
φ_3	Display$_1$	Pilot-1 sees the position of the Plane-2 .
φ_4	Display$_2$	Pilot-2 sees the position of the Plane-1.
φ_5	ATC Command	Initiation of transmission of "descend RA" to Plane-2.
φ_6	Pilot-2	Pilot-2 follows the ATC instruction and descends.
φ_7	Communication	Incorrect communication between pilots from Plane-1 and Plane-2 based on the received RA's.
φ_8	ADS-B-1	Initiation of transmission of position data to Plane-2.
φ_9	ADS-B-2	Initiation of transmission of position data to Plane-1.
φ_{10}	Verification	Pilot-1 and Pilot-2 check and verify the position of the respective aircraft and verify it with the position data displayed.

S_{p1}	p1 = T	$\boldsymbol{p1 = \varphi_1 \wedge \varphi_3 \wedge \neg\varphi_7 \wedge \varphi_8 \wedge \varphi_{10}}$
		df(TCAS-1) ∧ df(Display$_1$)∧ ~df(Communication) ∧ df(ADS-B-1) ∧ df(Verification)
		Output = *Position of Plane-2*

S_{p2}	p2 = T	$\boldsymbol{p2 = \varphi_2 \wedge \varphi_4 \wedge \neg\varphi_5 \wedge \neg\varphi_6 \wedge \neg\varphi_7 \wedge \varphi_9 \wedge \varphi_{10}}$
		df(TCAS-2) ∧ df(Display$_2$)∧ ~df(ATC Command) ∧ ~df(Pilot-2) ∧ ~df(Communication) ∧ df(ADS-B-2) ∧ df(Verification)
		Output = *Position of Plane-1*

S_c	c = T	$\boldsymbol{c = \varphi_1 \wedge \varphi_2 \wedge \neg\varphi_5 \wedge \neg\varphi_7}$
		df(TCAS-1) ∧ df(TCAS-2) ∧ ~df(ATC Command) ∧ ~df(Communication) ∧ df(Verification)
		Output = *Position of Plane-1 and Plane-2*

S_{a1}	$a1$ = T	$\boldsymbol{a1 = \varphi_8}$
		df(ADS-B-1)
		Output = *Position of Plane-2*

S_{a2}	$a2$ = T	$\boldsymbol{a2 = \varphi_9}$
		df(ADS-B-2)
		Output = *Position of Plane-1*

For each satellite being tracked, the receiver now knows where the satellite was at the time of transmission and it has determined the distance to the satellite when it was there. Using trilateration, a method of geometrically determining the position of an object, in a manner similar to triangulation, the receiver calculates its position.

The two security domains in this scenario are SD^C {controller Domain} and SD^P {pilot-1 and pilot-2 Domain}. By combining the valuation functions in SD^C and SD^P with respect to the invariants from ADS-B domain with respect to invariants from Equation 6.53 and Equation 6.54 in ADS-B domain,

$$S_c = \varphi_1 \wedge \varphi_2 \wedge \neg\varphi_5 \wedge \neg\varphi_7 \Rightarrow \nexists V^C_{\sim i} \tag{6.55}$$

Since the information received from the controller domain is faulty, the pilots cannot valuate for correctness of the position data in that domain.

$$S_p = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4 \wedge \neg\varphi_5 \wedge \neg\varphi_6 \wedge \neg\varphi_7 \wedge \varphi_8 \wedge \varphi_9 \wedge \varphi_{10} \Rightarrow \exists V^P_i \tag{6.56}$$

Since the information received from the pilot-1 and pilot-2 domain is not faulty, the pilots can valuate for correctness of the position data in that domain.

By combining Equation 6.55 and Equation 6.56,

$$MSDND(ES) = \exists w \in W\colon [w \vdash \Box f(S_c, S_p)] \wedge [w \models (\nexists V^C_{\sim i} \wedge \exists V^P_i)] \tag{6.57}$$

Therefore, the pilots can deduce that ATC controller instructed with wrong commands and pilots can follow the initial RAs issued by the TCAS system.

Hence, the system is *not Nondeducible secure* to the pilots as they can deduce the correctness of ATC commands.

The ProVerif code in APPENDIX I proves that this attack is *not Nondeducible secure* when ADS-B is used as an alternate information flow path. As can be interpreted from "RESULT not attacker(XY_Coord[]) is false", the TCAS is sending incorrect commands.

Table 6.17 presents the summary of attack scenarios and the corresponding invariants source used to identify the attack.

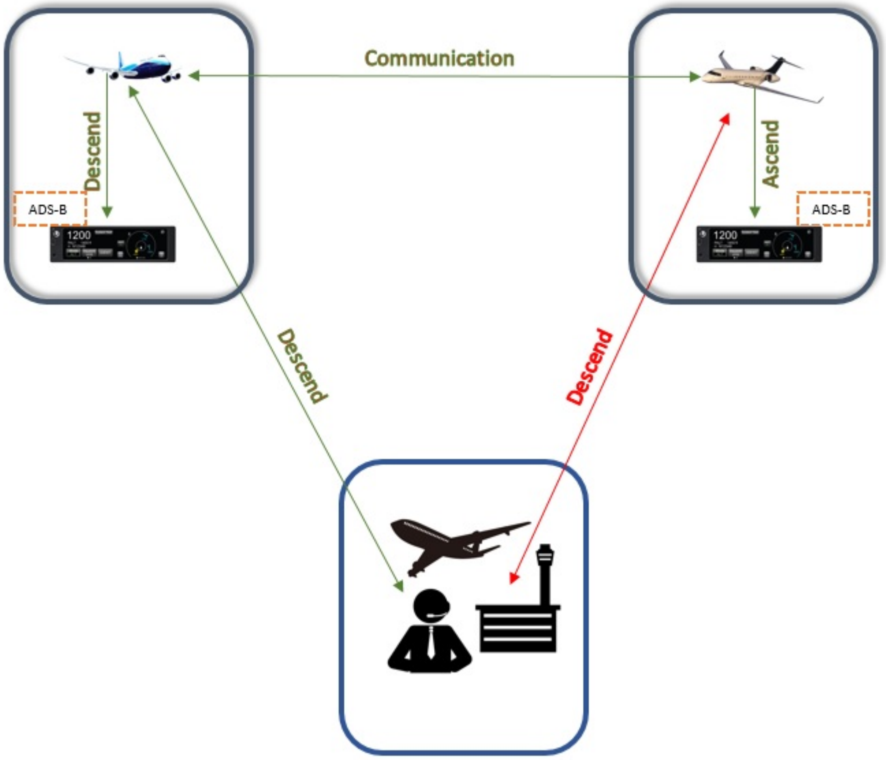

Figure 6.16. ADS-B Helps Identify The Incorrect Commands From ATC

Table 6.17. Summary Of The Attack Scenarios

Scenario	**MSDND Secure**	**Invariants Source**
Radar Transponder Failure	No	INS, ADS-B
Radar Transponder Failure and TCAS	No	INS, ADS-B
Altimeter Failure	No	INS, ADS-B
Pitot Static System Failure	No	Alternate static source, LRI
ADS-B Transmitter Failure	No	INS
ADS-B, INS and Attacker	No	TCAS
ADS-B and RF Interference	Yes	NA
ADS-B and Satellite Failure	Yes	NA
TCAS and ATC Controller	No	ADS-B

7. FUTURE WORK

The introduction of an automated, decentralized airspace system and the aircraft flying in that airspace system, leads to some unique CPS challenges and open problems. The air traffic control network will be used by manned and unmanned air vehicles in the shared airspace and additionally be used for distribution of large volumes of potentially non-safety critical data, e.g., traffic information and updates from ground systems. Furthermore, with growth of programs such as "fly-by-wireless" [10], an on-board wireless network communicates advisory and eventually time-critical data, will be increasingly an option for aviation. This results in a complex system with a huge number of information flow paths between various systems in the aircraft. The MSDND model must exhibit deducibility for time critical operations under above specified conditions, e.g., in terms of deducing the attack when the traffic is high. Hence the performance of MSDND in mixed traffic loads must be studied for future aircraft. Furthermore, risks from tight cyber-physical integration must be assessed for their potential impact on physical behavior and performance gains of future aircraft and airspace systems.

A strategic and generalized way to partition the security domains is needed. While in this thesis, the assignment of security domains to physical domains seems natural, it is expected that this will not always be the case. A formal proof is needed that quantifies the security domains and assesses the number of information flow paths that can be corrupted and still maintain deducibility. In the future, systems, equipment and components will have their own intelligence and communication abilities, which makes the cyber-physical interactions more complex. Focus can be laid on building a fully automated tool to automate the MSDND analysis by partitioning the CPS into security domains, devise attack scenarios and provide alternatives to overcome bottlenecks in the overall system.

There are technical challenges associated with tightening the cyber-physical integration within aircraft and between aircraft and off-board systems. Most aviation standards do not yet cover cyber-physical threats. Several aviation regulatory agencies have started creating regulations (e.g., Special Conditions) to address cyber security (e.g., network security) and physical security (e.g., onboard RF interference) concerns [10]. Physical risks from bad weather, icing, winds, solar flares, can also impact cyber performance (e.g., GPS outage due to solar flares).

Future work also includes extending the MSDND model to include entire aviation systems, as well as identifying and assessing aviation cyber-physical risks.

8. CONCLUSION

Viewing the security domains as single high and low domains is not adequate as it does not cover all the possible combinations of the states. The MSDND model is used to analyze the information flow paths in complex CPS and this model is applied to a specific CPS, air traffic surveillance systems, with quick response times. This model fits the complex CPS better compared to other models, because it considers all the possible combinations of states and applies MSDND to check for the deducibility. The MSDND model shows where the vulnerabilities lie and shows where additional inputs or invariants are needed to mitigate the vulnerabilities.

This thesis considers possible attacks and malfunctions in the air traffic surveillance systems. The benefits of tightening the cyber-physical interactions of airborne computer systems, however, is at the cost of exposure to potential vulnerabilities. The thesis noted that the CPS can be partitioned into multiple overlapping security domains and the information flow paths between these domains can be fundamentally viewed as a security function for mitigating vulnerability exposure, but these information flow paths are also subject to failure and intentional disruption. A system sending information to other systems has to be evaluated for effectiveness, including failure impact and required reliability. This thesis is aimed at evaluating the security domains and the associated paths, using various attack scenarios to identify the attack. Given an attack scenario and the source of the attack, MSDND makes use of the invariants present in the system to provide an alternative to avoid the attack.

This thesis provided a CPS vision for future aviation information systems of aircraft. The presented MSDND analysis offers new insights on the security analysis of aircraft and airspace systems as well as the impact of these risks on the emerging cyber-physical interactions in aerospace.

APPENDIX A

RADAR TRANSPONDER FAILURE

CODE:

```
(* Defining user-defined data type as Coordinates*)
type Coord.

(* Variables, Names and Types *)
(* Making channel c private for secure transfer of data*)
free c:channel [private].
free ch:channel.
free Accept:bitstring [private].
free Request:bitstring [private].
free XY_Coord:Coord.

(*query to see if attacker can obtain any of the messages*)
query attacker(Accept).
query attacker(Request).
query attacker(XY_Coord).

(*Process1 at ATC *)
let ATC=
    (*Request sent to only Flight A*)
        out(ch, Request);
    (* Coordinates are received from Flight A, and are stored
    in XY_Location *)
        in(c, XY_Location:Coord);
        if XY_Location = XY_Coord then
    (* Coordinates are accepted *)
        out(c, Accept).

(*Process2 at Flight A TCAS*)
let TCAS_FlightA=
    (* Request is taken by Flight A *)
        in(c,x_Request:bitstring);
        if x_Request=Request then
    (* Coordinates are sent as response *)
        out(ch, XY_Coord).

(*Parallel Composition of ATC and Flight A TCAS processes*)
process
         ((! ATC) | (! TCAS_FlightA))
```

OUTPUT:

```
-- Query not attacker(XY_Coord[])
Completing...
Starting query not attacker(XY_Coord[])
goal reachable: attacker(XY_Coord[])
RESULT not attacker(XY_Coord[]) is false.
-- Query not attacker(Request[])
Completing...
Starting query not attacker(Request[])
goal reachable: attacker(Request[])
RESULT not attacker(Request[]) is false.
-- Query not attacker(Accept[])
```

```
Completing...
Starting query not attacker(Accept[])
RESULT not attacker(Accept[]) is true.
```

APPENDIX B

RADAR TRANSPONDER FAILURE AND TCAS

CODE:

```
(* Defining user-defined data type as Coordinates*)
type Coord.

(* Variables, Names and Types *)
(* Making channel c private for secure transfer of data*)
free c:channel [private].
free ch:channel.
free Accept:bitstring [private].
free Request:bitstring [private].
free XY_Coord:Coord.

(*query to see if attacker can obtain any of the messages*)
query attacker(Accept).
query attacker(Request).
query attacker(XY_Coord).

(* Process1 at ATC *)
let ATC=
    (*Request sent to only Flight A*)
        out(ch, Request);
    (* Coordinates are received from Flight A, and are
    stored in XY_Location *)
        in(c, XY_Location:Coord);
        if XY_Location = XY_Coord then
    (* Coordinates are accepted *)
        out(c, Accept).

(* Process2 at Flight A TCAS*)
let TCAS_FlightA=
    (* Request is taken by Flight A *)
        in(c,x_Request:bitstring);
        if x_Request=Request then
    (* Coordinates are sent as response *)
        out(ch, XY_Coord).

(* Process3 at Flight B TCAS*)
let TCAS_FlightB=
    (* Request is taken by Flight B *)
        in(c,x_Request:bitstring);
        if x_Request=Request then
    (* Coordinates are sent as response *)
        out(ch, XY_Coord).

process
         ((! ATC) | (! TCAS_FlightA) | (! TCAS_FlightB))
```

OUTPUT:

```
-- Query not attacker(XY_Coord[])
Completing...
Starting query not attacker(XY_Coord[])
goal reachable: attacker(XY_Coord[])
```

```
RESULT not attacker(XY_Coord[]) is false.
-- Query not attacker(Request[])
Completing...
Starting query not attacker(Request[])
goal reachable: attacker(Request[])
RESULT not attacker(Request[]) is false.
-- Query not attacker(Accept[])
Completing...
Starting query not attacker(Accept[])
RESULT not attacker(Accept[]) is true.
```

APPENDIX C

ALTIMETER FAILURE

CODE:

```
(* Defining user-defined data type as Coordinates*)
type Coord.

(* Variables, Names and Types *)
(* Making channel c private for secure transfer of data*)
free c:channel [private].
free ch:channel.
free Accept:bitstring [private].
free Request:bitstring [private].
free XY_Coord:Coord [private].

(*query to see if attacker can obtain any of the messages*)
query attacker(Accept).
query attacker(Request).
query attacker(XY_Coord).

(* TCAS Altimeter process1 at FlightA *)
let TCAS_Alti_FlightA=
        (*Request sent to only Flight B*)
        out(c, Request);
        (* Coordinates received from Flight B*)
        in(c, XY_Location:Coord);
        if XY_Location = XY_Coord then
        (* Coordinates are accepted *)
        out(ch, Accept).
```

```
(* TCAS process2 at Flight B *)
let TCAS_FlightB=
        (* Request is taken by Flight B *)
        in(c,x_Request:bitstring);
        if x_Request=Request then
        (* Coordinates are sent as response *)
        out(c, XY_Coord).


process
         ((! TCAS_Alti_FlightA) | (! TCAS_FlightB))
```

OUTPUT:

```
-- Query not attacker(XY_Coord[])
Completing...
Starting query not attacker(XY_Coord[])
RESULT not attacker(XY_Coord[]) is true.
-- Query not attacker(Request[])
Completing...
Starting query not attacker(Request[])
RESULT not attacker(Request[]) is true.
-- Query not attacker(Accept[])
Completing...
Starting query not attacker(Accept[])
goal reachable: attacker(Accept[])
RESULT not attacker(Accept[]) is false.
```

APPENDIX D

PITOT STATIC SYSTEM FAILURE

CODE:

```
type netValue.
type P.

(* Variables, Names and Types *)
free c:channel.
free c_phy:channel [private].
free ch:channel [private].
free Status:bitstring.free Request:bitstring.
free Response:bitstring [private].
free Select:bitstring [private].
free Safe:bitstring [private].
free Unsafe:bitstring [private].free Set:bitstring [private].
free p1,p2:P [private].
free p3:P.free delta1, delta2, delta3:netValue [private].
(* Invariant process *)
fun Inv(P):netValue.reduc forall p:P,
value:netValue; pos(Inv(p), value)= value.

(* Pitot Static System process *)
let Pitot =
        out(c, Request);
        in(ch, x_Response:bitstring);
        if x_Response = Response then
        out(ch, choice[Safe, Unsafe]);
        in(ch, x_Safe:bitstring);
        if x_Safe = Safe then
        out(c_phy, Inv(p1));
        if x_Safe = Unsafe then
        out(c, Inv(p1));
        let y=pos(Inv(p1), delta1) in
        out(ch, Set).

(* TCAS System process *)
        let TCAS =
        in(c,x_Request:bitstring);
        if x_Request=Request then
        out(ch, Response);
        in(ch, x_Select:bitstring);
        if x_Select=Select then
        out(ch, Safe);
        out(c_phy, Inv(p2));
        let y=pos(Inv(p2), delta2) in
        out(ch, Set).

(* Alternate Static Source process *)
        let Alternate =
        out(c,Status);
        in(c,x_Request:bitstring);
        if x_Request=Request then
        out(ch, Response);
        in(ch, x_Select:bitstring);
        if x_Select=Select then
```

```
          out(ch, Safe);
          out(c_phy, Inv(p3));
          let y=pos(Inv(p3), delta3) in
          out(ch, Set).

process ((! TCAS) | (! Alternate))
```

OUTPUT:

```
-- Observational equivalence
Completing...
RESULT Observational equivalence is true (bad not derivable).
```

APPENDIX E

ADS-B TRANSMITTER FAILURE

CODE:

```
type netValue.
type P.

(* Variables, Names and Types *)
free c:channel.
free c_phy:channel [private].
free ch:channel [private].
free Status:bitstring.free Request:bitstring.
free Response:bitstring [private].
free Select:bitstring [private].
free Safe:bitstring [private].
free Unsafe:bitstring [private].free Set:bitstring [private].
free p1,p2:P [private].
free p3:P.free delta1, delta2, delta3:netValue [private].

(* Invariant process *)
fun Inv(P):netValue.reduc forall p:P, value:netValue;
pos(Inv(p), value)= value.



(* Aircraft1 process *)
let Aircraft_1 =
        out(c, Status);
        out(c, Request);
        in(ch, x_Response:bitstring);
        if x_Response = Response then
        out(ch, choice[Safe, Unsafe]);
        in(ch, x_Safe:bitstring);
        if x_Safe = Safe then   out(c_phy, Inv(p1));
        if x_Safe = Unsafe then
        out(c, Inv(p1));
        let y=pos(Inv(p1), delta1) in
        out(ch, Set).


(* Aircraft2 process *)
        let Aircraft_2 =
        out(c,Status);
        in(c,x_Request:bitstring);
        if x_Request=Request then
        out(ch, Response);
        in(ch, x_Select:bitstring);
        if x_Select=Select then
        out(ch, Safe);
        out(c_phy, Inv(p2));
        let y=pos(Inv(p2), delta2) in
        out(ch, Set).


(* Aircraft3 process *)
        let Aircraft_3 =
        out(c,Status);
```

```
    in(c,x_Request:bitstring);
    if x_Request=Request then
    out(ch, Response);
    in(ch, x_Select:bitstring);
    if x_Select=Select then
    out(ch, Unsafe);
    out(c_phy, Inv(p3));
    let y=pos(Inv(p3), delta3) in
    out(ch, Set).

process ((! Aircraft_2) | (! Aircraft_3))
```

OUTPUT:

```
-- Observational equivalence
Completing...
RESULT Observational equivalence is true (bad not derivable).
```

APPENDIX F

ADS-B, INS AND ATTACKER

CODE:

```
(* Defining user-defined data type as Coordinates*)
type Coord.

(* Variables, Names and Types *)
(* Making channel c private for secure transfer of data*)
free c:channel [private].
free Accept:bitstring [private].
free Request:bitstring [private].

free XY_Coord_Node_1:Coord.
free XY_Coord_Node_2:Coord [private].
free XY_Coord_Node_3:Coord [private].
free XY_Coord_Node_4:Coord [private].
free XY_Coord_Node_5:Coord [private].
free XY_Coord_Node_6:Coord [private].
free XY_Coord_Node_7:Coord [private].
free XY_Coord_Node_8:Coord [private].
free XY_Coord_Node_9:Coord [private].

(*query to see if attacker can obtain any of the messages*)
query attacker(Accept).
query attacker(Request).
query attacker(XY_Coord_Node_1).
query attacker(XY_Coord_Node_2).
query attacker(XY_Coord_Node_3).
query attacker(XY_Coord_Node_4).
query attacker(XY_Coord_Node_5).
query attacker(XY_Coord_Node_6).
query attacker(XY_Coord_Node_7).
query attacker(XY_Coord_Node_8).
query attacker(XY_Coord_Node_9).

(* Comm process1 at FlightA *)
let Comm_FlightA=
    (*Request sent to only Flight B*)
        out(c, Request);
        in(c, XY_Location:Coord);
        if XY_Location = XY_Coord_Node_1 || XY_Location =
        XY_Coord_Node_2 || XY_Location = XY_Coord_Node_3 ||
        XY_Location = XY_Coord_Node_4 || XY_Location =
        XY_Coord_Node_5 || XY_Location = XY_Coord_Node_6 ||
        XY_Location = XY_Coord_Node_7 || XY_Location =
        XY_Coord_Node_8 || XY_Location = XY_Coord_Node_9 then
        out(c, Accept).

(* Process2 at Node_1*)
let Quad_Node_1=
        in(c,x_Request:bitstring);
        if x_Request=Request then
        out(c, XY_Coord_Node_1).

(* process2 at Node_2 *)
```

```
let Quad_Node_2=
        in(c,x_Request:bitstring);
        if x_Request=Request then
        out(c, XY_Coord_Node_2).

(* process2 at Node_3 *)
let Quad_Node_3=
    in(c,x_Request:bitstring);
        if x_Request=Request then
        out(c, XY_Coord_Node_3).

(* process2 at Node_4 *)
let Quad_Node_4=
        in(c,x_Request:bitstring);
        if x_Request=Request then
        out(c, XY_Coord_Node_4).

(* process2 at Node_5 *)
let Quad_Node_5=
        in(c,x_Request:bitstring);
        if x_Request=Request then
        out(c, XY_Coord_Node_5).

(* process2 at Node_6 *)
let Quad_Node_6=
        in(c,x_Request:bitstring);
        if x_Request=Request then
        out(c, XY_Coord_Node_6).

(* process2 at Node_7*)
let Quad_Node_7=
        in(c,x_Request:bitstring);
        if x_Request=Request then
        out(c, XY_Coord_Node_7).

(* process2 at Node_8 *)
let Quad_Node_8=
        in(c,x_Request:bitstring);
        if x_Request=Request then
        out(c, XY_Coord_Node_8).

(* process2 at Node_9 *)
let Quad_Node_9=
        in(c,x_Request:bitstring);
        if x_Request=Request then
        out(c, XY_Coord_Node_9).

process
        ((!Comm_FlightA) |
        (!Quad_Node_1) | (!Quad_Node_2) | (!Quad_Node_3) |
        (!Quad_Node_4) | (!Quad_Node_5) | (!Quad_Node_6) |
        (!Quad_Node_7) | (!Quad_Node_8) | (!Quad_Node_9))
```

OUTPUT:

```
-- Query not attacker(XY_Coord_Node_9[])
Completing...
Starting query not attacker(XY_Coord_Node_9[])
RESULT not attacker(XY_Coord_Node_9[]) is true.
-- Query not attacker(XY_Coord_Node_8[])
Completing...
Starting query not attacker(XY_Coord_Node_8[])
RESULT not attacker(XY_Coord_Node_8[]) is true.
-- Query not attacker(XY_Coord_Node_7[])
Completing...
Starting query not attacker(XY_Coord_Node_7[])
RESULT not attacker(XY_Coord_Node_7[]) is true.
-- Query not attacker(XY_Coord_Node_6[])
Completing...
Starting query not attacker(XY_Coord_Node_6[])
RESULT not attacker(XY_Coord_Node_6[]) is true.
-- Query not attacker(XY_Coord_Node_5[])
Completing...
Starting query not attacker(XY_Coord_Node_5[])
RESULT not attacker(XY_Coord_Node_5[]) is true.
-- Query not attacker(XY_Coord_Node_4[])
Completing...
Starting query not attacker(XY_Coord_Node_4[])
RESULT not attacker(XY_Coord_Node_4[]) is true.
-- Query not attacker(XY_Coord_Node_3[])
```

```
Completing...
Starting query not attacker(XY_Coord_Node_3[])
RESULT not attacker(XY_Coord_Node_3[]) is true.
-- Query not attacker(XY_Coord_Node_2[])
Completing...
Starting query not attacker(XY_Coord_Node_2[])
RESULT not attacker(XY_Coord_Node_2[]) is true.
-- Query not attacker(XY_Coord_Node_1[])
Completing...
Starting query not attacker(XY_Coord_Node_1[])
goal reachable: attacker(XY_Coord_Node_1[])
RESULT not attacker(XY_Coord_Node_1[]) is false.
-- Query not attacker(Request[])
Completing...
Starting query not attacker(Request[])
RESULT not attacker(Request[]) is true.
-- Query not attacker(Accept[])
Completing...
Starting query not attacker(Accept[])
RESULT not attacker(Accept[]) is true.
```

APPENDIX G

ADS-B AND RF INTERFERENCE

CODE:

```
type netValue.
type P.

(* Variables, Names and Types *)
free c:channel.
free c_phy:channel [private].
free ch:channel [private].
free Status:bitstring.
free Request:bitstring.
free Response:bitstring [private].
free Select:bitstring [private].
free Safe:bitstring [private].
free Unsafe:bitstring [private].
free Set:bitstring [private].
free p1,p2:P [private].
free delta1, delta2:netValue [private].

(* Invariant process *)
fun Inv(P):netValue.reduc forall p:P,
value:netValue; pos(Inv(p), value)= value.

(* Aircraft1 process *)
let Aircraft_1 =
        out(c, Status);
        out(c, Request);
```

```
          in(ch, x_Response:bitstring);
          if x_Response = Response then
          out(ch, choice[Safe, Unsafe]);
          in(ch, x_Safe:bitstring);
          if x_Safe = Safe then   out(c_phy, Inv(p1));
          if x_Safe = Unsafe then
          out(c, Inv(p1));
          let y=pos(Inv(p1), delta1) in
          out(ch, Set).

(* Aircraft2 process *)
          let Aircraft_2 =
          out(c,Status);
          in(c,x_Request:bitstring);
          if x_Request=Request then
          out(ch, Response);
          in(ch, x_Select:bitstring);
          if x_Select=Select then
          out(ch, Safe);
          out(c_phy, Inv(p2));
          let y=pos(Inv(p2), delta2) in
          out(ch, Set).

process ((! Aircraft_2) | (! Aircraft_1))
```

OUTPUT:

```
-- Observational equivalence
```

```
Termination warning: v_237 <> v_238 && attacker2(v_236,v_237)
&& attacker2(v_236,v_238) -> bad
Selecting 0
Termination warning: v_240 <> v_241 && attacker2(v_240,v_239)
&& attacker2(v_241,v_239) -> bad
Selecting 0
Completing...
Termination warning: v_237 <> v_238 && attacker2(v_236,v_237)
&& attacker2(v_236,v_238) -> bad
Selecting 0
Termination warning: v_240 <> v_241 && attacker2(v_240,v_239)
&& attacker2(v_241,v_239) -> bad
Selecting 0
goal reachable: bad
RESULT Observational equivalence cannot be proved
(bad derivable).
Looking for simplified processes ...
No simplified process found
```

APPENDIX H

ADS-B AND SATELLITE FAILURE

CODE:

```
type netValue.
type P.

(* Variables, Names and Types *)
free c:channel.
free c_phy:channel [private].
free ch:channel [private].
free Status:bitstring.free Request:bitstring.
free Response:bitstring [private].
free Position1:bitstring [private].
free Position2:bitstring [private].
free Select:bitstring [private].
free Safe:bitstring [private].
free Unsafe:bitstring [private].
free Set:bitstring [private].
free p1,p2:P [private].
free delta1, delta2:netValue [private].

(* Invariant process *)
fun Inv(P):netValue.reduc forall p:P,
value:netValue; pos(Inv(p), value)= value.

(* Satellite process *)
        let Satellite=
        out(c,Status);
        out(ch, Position1);
        out(ch, Position2).

(* Aircraft1 process *)
let Aircraft_1 =
        out(c, Status);
        out(c, Request);
        in(ch, x_Response:bitstring);
        if x_Response = Position1 then
        out(ch, choice[Safe, Unsafe]);
        in(ch, x_Safe:bitstring);
        if x_Safe = Safe then
        out(c_phy, Inv(p1));
        if x_Safe = Unsafe then
        out(c, Inv(p1));
        let y=pos(Inv(p1), delta1) in
        out(ch, Set).

(* Aircraft2 process *)
        let Aircraft_2 =
        out(c,Status);
        out(c, Request);
        in(c,x_Response:bitstring);
        if x_Response=Position2 then
        out(ch, choice[Safe, Unsafe]);
        in(ch, x_Safe:bitstring);
        if x_Safe = Safe then
        out(c_phy, Inv(p2));
        if x_Safe = Unsafe then
        out(c, Inv(p2));
        let y=pos(Inv(p2), delta2) in
        out(ch, Set).

process ((! Satellite) | (! Aircraft_2) | (! Aircraft_1))
```

OUTPUT:

```
-- Observational equivalence
Termination warning: v_239 <> v_240 && attacker2(v_238,v_239)
&& attacker2(v_238,v_240) -> bad
Selecting 0
Termination warning: v_242 <> v_243 && attacker2(v_242,v_241)
&& attacker2(v_243,v_241) -> bad
Selecting 0
Completing...
Termination warning: v_239 <> v_240 && attacker2(v_238,v_239)
&& attacker2(v_238,v_240) -> bad
Selecting 0
Termination warning: v_242 <> v_243 && attacker2(v_242,v_241)
&& attacker2(v_243,v_241) -> bad
Selecting 0
goal reachable: bad
Could not find a trace corresponding to this derivation.
RESULT Observational equivalence cannot be proved
(bad derivable).
```

APPENDIX I

TCAS AND ATC CONTROLLER

CODE:

```
(* Defining user-defined data type as Coordinates*)
type Coord.

(* Variables, Names and Types *)
(* Making channel c private for secure transfer of data*)
free c:channel [private].
free ch:channel.
free Accept:bitstring [private].
free Request:bitstring [private].
free XY_Coord:Coord.

(*query to see if attacker can obtain any of the messages*)
query attacker(Accept).
query attacker(Request).
query attacker(XY_Coord).

(* Process1 at ATC *)
let ATC=
    (*Request sent to only Flight A*)
        out(ch, Request);
    (* Coordinates are received from Flight A, and are
    stored in XY_Location *)
        in(c, XY_Location:Coord);
        if XY_Location = XY_Coord then
    (* Coordinates are accepted *)
```

```
        out(c, Accept).

(* Process2 at Flight A TCAS*)
let TCAS_FlightA=
    (* Request is taken by Flight A *)
        in(c,x_Request:bitstring);
        if x_Request=Request then
    (* Coordinates are sent as response *)
        out(ch, XY_Coord).

process
         ((! ATC) | (! TCAS_FlightA))
```

OUTPUT:

```
-- Query not attacker(XY_Coord[])
Completing...
Starting query not attacker(XY_Coord[])
goal reachable: attacker(XY_Coord[])
RESULT not attacker(XY_Coord[]) is false.
-- Query not attacker(Request[])
Completing...
Starting query not attacker(Request[])
goal reachable: attacker(Request[])
RESULT not attacker(Request[]) is false.
-- Query not attacker(Accept[])
```

```
Completing...
Starting query not attacker(Accept[])
RESULT not attacker(Accept[]) is true.
```

BIBLIOGRAPHY

[1] FA FAA. Pilot's Handbook of Aeronautical Knowledge. *Washington: Government Printing Office*, 2009.

[2] J Wiley. FAA test challenges in the 21st century. *ITEA Journal*, 29:117–119, 2008.

[3] Sudhakar Shetty. System of systems design for worldwide commercial aircraft networks. *Proceedings of ICAS, International Council of the Aeronautical Sciences*, 8 (1):2008, 2008.

[4] Krishna Sampigethaya and Radha Poovendran. Aviation cyber–physical systems: Foundations for future aircraft and air transport. *Proceedings of the IEEE*, 101(8): 1834–1855, 2013.

[5] Krishna Sampigethaya, Radha Poovendran, and Linda Bushnell. Secure operation, control, and maintenance of future e-enabled airplanes. *Proceedings of the IEEE*, 96 (12):1992–2007, 2008.

[6] Chris A Wargo and Chris Dhas. Security consideratiolis for the e-enabled aircraft. In *Aerospace Conference, 2003. Proceedings. 2003 IEEE*, volume 4, pages 4_1533–4_1550. IEEE, 2003.

[7] Krishna Sampigethaya, Radha Poovendran, Sudhakar Shetty, Terry Davis, and Chuck Royalty. Future e-enabled aircraft communications and security: The next 20 years and beyond. *Proceedings of the IEEE*, 99(11):2040–2055, 2011.

[8] A. Nourian and S. Madnick. A Systems Theoretic Approach to the Security Threats in Cyber Physical Systems Applied to Stuxnet. *IEEE Transactions on Dependable and Secure Computing*, PP(99):1–1, 2015. ISSN 1545-5971.

[9] Radha Poovendran, Krishna Sampigethaya, Sandeep Kumar S Gupta, Insup Lee, K Venkatesh Prasad, David Corman, and James L Paunicka. Special issue on cyber-physical systems [scanning the issue]. *Proceedings of the IEEE*, 100(1):6–12, 2012.

[10] Chuck Royalty. Cyber Security for Aeronautical Networked Platforms-What does it mean to me in commercial aviation design? In *Infotech@ Aerospace*, 2012.

[11] Alvaro A Cárdenas, Saurabh Amin, Zong-Syun Lin, Yu-Lun Huang, Chi-Yen Huang, and Shankar Sastry. Attacks against process control systems: risk assessment, detection, and response. In *Proceedings of the 6th ACM symposium on information, computer and communications security*, pages 355–366. ACM, 2011.

[12] Alvaro A Cárdenas, Saurabh Amin, and Shankar Sastry. Research Challenges for the Security of Control Systems. In *HotSec*, 2008.

[13] Vinay M Igure, Sean A Laughter, and Ronald D Williams. Security issues in SCADA networks. *Computers & Security*, 25(7):498–506, 2006.

[14] E. Johansson, T. Sommestad, and M. Ekstedt. Issues of cyber security in SCADA-systems - On the importance of awareness. In *CIRED 2009 - 20th International Conference and Exhibition on Electricity Distribution - Part 1*, pages 1–4, June 2009. doi: 10.1049/cp.2009.1099.

[15] Henrik Christiansson and Eric Luiijf. Creating a european scada security testbed. In *International Conference on Critical Infrastructure Protection*, pages 237–247. Springer, 2007.

[16] Himanshu Khurana, Mark Hadley, Ning Lu, and Deborah A Frincke. Smart-grid security issues. *IEEE Security & Privacy*, 8(1), 2010.

[17] Brian Krebs. Cyber incident blamed for nuclear power plant shutdown. *Washington Post, June*, 5:2008, 2008.

[18] Andrei Sabelfeld and Andrew C Myers. Language-based information-flow security. *IEEE Journal on selected areas in communications*, 21(1):5–19, 2003.

[19] Andrew C Myers, Lantian Zheng, Steve Zdancewic, Stephen Chong, and Nathaniel Nystrom. Jif: Java information flow. *Software release. Located at http://www. cs. cornell. edu/jif*, 2005, 2001. [Online: date accessed 20-May-2017].

[20] Vincent Simonet and Inria Rocquencourt. Flow Caml in a nutshell. In *Proceedings of the first APPSEM-II workshop*, pages 152–165. Nottingham, United Kingdom, 2003.

[21] François Pottier and Vincent Simonet. Information flow inference for ML. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 25(1):117–158, 2003.

[22] David Sutherland. A model of Information. In *Proc. 9th National Computer Security Conference*, pages 175–183. DTIC Document, 1986.

[23] John McLean. Security Models and Information Flow. In *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on*, pages 180–187. IEEE, 1990.

[24] G. Howser and B. McMillin. A Multiple Security Domain Model of a Drive-by-Wire System. In *2013 IEEE 37th Annual Computer Software and Applications Conference*, pages 369–374, July 2013. doi: 10.1109/COMPSAC.2013.62.

[25] Bruno Blanchet. Vérification automatique de protocoles cryptographiques: modele formel et modele calculatoire. *Mémoire d'habilitationa diriger des recherches, Université Paris-Dauphine*, 2008.

[26] Martín Abadi and Bruno Blanchet. Analyzing security protocols with secrecy types and logic programs. *Journal of the ACM (JACM)*, 52:102–146, 2005.

[27] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *ACM Sigplan Notices*, volume 36, pages 104–115. ACM, 2001.

[28] B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proceedings. 14th IEEE Computer Security Foundations Workshop, 2001.*, pages 82–96, 2001. doi: 10.1109/CSFW.2001.930138.

[29] Danny Dolev and Andrew Yao. On the security of public key protocols. *IEEE Transactions on information theory*, 29(2):198–208, 1983.

[30] Martín Abadi and Andrew D Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proceedings of the 4th ACM conference on Computer and communications security*, pages 36–47. ACM, 1997.

[31] Bruno Blanchet. ProVerif automatic cryptographic protocol verifier user manual. *CNRS, Departement dInformatique, Ecole Normale Superieure, Paris*, 2005.

[32] Chien-Ying Chen, AmirEmad Ghassami, Sibin Mohan, Negar Kiyavash, Rakesh B. Bobba, Rodolfo Pellizzoni, and Man-Ki Yoon. A Reconnaissance Attack Mechanism for Fixed-Priority Real-Time Systems. *CoRR*, abs/1705.02561, 2017. URL `http://arxiv.org/abs/1705.02561`.

[33] Joseph Watson. *Aerospace Navigation Systems*. John Wiley & Sons, 2016.

[34] Pangun Park and Claire Tomlin. Investigating Communication Infrastructure of Next Generation Air Traffic Management. In *Proceedings of the 2012 IEEE/ACM Third International Conference on Cyber-Physical Systems*, pages 35–44. IEEE Computer Society, 2012.

[35] Jeppesen Sanderson. Guided Flight Discovery: Private Pilot. *Pat Willits*, 2004.

[36] David Evans. Safety: Maintenance Snafu with Static Ports. *Avionics Magazine. Retrieved*, pages 04–25, 2008.

[37] Sonex Foundation. Sonex Builders & Pilots Foundation. http://www.sonexfoundation.com/Lift_Reserve_Indicator.html, 2016. [Online: date accessed 20-May-2017].

[38] National Aeronautics and Space Administration. Aviation Navigation. http://virtualskies.arc.nasa.gov/navigation/4.html. [Online: date accessed 20-May-2017].

[39] Paul Marks. Air traffic system vulnerable to cyber attack. *New Scientist*, 211(2829): 22–23, 2011.

[40] H Kelly. Researcher: New air traffic control system is hackable. *Cable News Network (CNN), Jul*, 2012.

[41] A Greenberg. Next-gen air traffic control vulnerable to hackers spoofing planes out of thin air. *Forbes Magazine. Retrieved September*, 10, 2012.

[42] K Zetter. Air traffic controllers pick the wrong week to quit using radar. *Wired, July*, 2012.

[43] Andrei Costin and Aurélien Francillon. Ghost in the Air (Traffic): On insecurity of ADS-B protocol and practical attacks on ADS-B devices. *Black Hat USA*, pages 1–12, 2012.

[44] Brad Haines. Hacker+ airplanes= no good can come of this. *Confidence X*, 2012.

[45] Righter Kunkel. Air traffic control insecurity 2.0. *Proc. DefCon*, 18, 2010.

[46] Matthias Schäfer, Vincent Lenders, and Ivan Martinovic. Experimental analysis of attacks on next generation air traffic communication. In *International Conference on Applied Cryptography and Network Security*, pages 253–271. Springer, 2013.

[47] P. R. Dunaka and B. McMillin. Cyber-Physical Security of a Chemical Plant. In *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)*, pages 33–40, Jan 2017. doi: 10.1109/HASE.2017.23.

[48] Robert Büssow, Robert Geisler, Wolfgang Grieskamp, and Marcus Klar. The μSZ Notation–Version 1.0. 1997.

VITA

Anusha Thudimilla was born in Hyderabad, India. She earned a Bachelors of Technology from Mahaveer Institute of Science and Technology, JNTU-H majoring in "Information Technology" in May 2012. She then worked with Thomson Reuters as an Associate Developer in Hyderabad, India from September 2012 to September 2013. Her work was mainly focused on creating web pages using front-end technologies. She later worked with Infosys Limited as a Systems Engineer for the Citi Bank client in chennai, India from September 2013 to June 2015. Her work was mainly focused on developing and maintaining COBOL and java applications.

She received her Master's degree in Computer Science from Missouri University of Science and Technology in December of 2017. While there she greatly enjoyed her work as a research assistant to Dr. Bruce McMillin for two years. Anusha presented her research work 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 1st prize at 18th IEEE International Symposium on High Assurance Systems Engineering (HASE) 2017 under student paper presentation competition. She also won 3rd place at 16th Annual CS Awards Banquet and Research Poster Showcase held by Missouri S&T Computer Science department.