Spring 2014

# Multiple security domain nondeducibility in cyber-physical systems

Gerry Wayne Howser



Department: Computer Science

# MULTIPLE SECURITY DOMAIN NONDEDUCIBILITY IN CYBER-PHYSICAL SYSTEMS

by

GERRY WAYNE HOWSER

A DISSERTATION

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

in Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

2014

Dr. Bruce M. McMillin, Advisor
Dr. Sajal Das, Computer Science
Dr. Wei Jiang, Computer Science
Dr. Daniel Tauritz, Computer Science
Dr. Sahra Sedigh Sarvestani, Electrical and Computer Engineering

## ABSTRACT

Cyber-Physical Systems (CPS) present special problems for security. This dissertation examines the cyber security problem, the physical security problem, the security problems presented when cyber systems and physical systems are intertwined, and problems presented by the fact that CPS leak information simply by being observed. The issues presented by applying traditional cyber security to CPS are explored and some of the shortcomings of these models are noted. Specific models of a "drive-by-wire" automobile connected to a road side assistance network, a "Stuxnet type" attack, the smart grid, and others are presented in detail.

The lack of good tools for CPS security is addressed in part by the introduction of a new model, Multiple Security Domains Nondeducibility over an Event System, or MSDND(ES). The drive-by-wire automobile is studied to show how MSDND(ES) is applied to a system that traditional security models do not describe well.

The issue of human trust in inherently vulnerable CPS with embedded cyber monitors, is also explored. A Stuxnet type attack on a CPS is examined using both MSDND(ES) and Belief, Information acquisition, and Trust (BIT) logic to provide a clear and precise method to discuss issues of trust and belief in monitors and electronic reports. To show these techniques, the electrical smart grid as envisioned by the Future Renewable Electric Energy Delivery and Management Systems Center (FREEDM) project is also modeled.

Areas that may lead to the development of additional tools are presented as possible future work to address the fact: CPS are different and require different models and tools to understand.

## ACKNOWLEDGMENTS

I would like to thank my advisor, friend, and co-author Bruce McMillin for his guidance and inspiration. Bruce has been a wonderful mentor. Working with him on publications has been enjoyable. I would also like to thank my PhD committee: Sajal Das, Wei Jiang, Sahra Sedigh Sarvestani, and Daniel Tauritz. I appreciate the support from all of you. While he was not a member of my committee, I would like to thank Sujeet Shenoi for his excellent suggestions and support on the first paper in this dissertation, *Modeling and reasoning about the security of drive-by-wire automobile systems.*

The support I received from Missouri University of Science and Technology, MS&T, was a key factor in my decision to complete my doctorate. The support of the Office of the Chancellor was important in allowing me the time to pursue my research. Being named Chancellor's Fellow in Computer Science provided validation of my research and inspired me. I also received support from the Computer Science Department as a teaching assistant and from the Council of Graduate Students in the form of a travel grant. The support is very much appreciated.

I also want to thank all the people the Computer Science Department at MS&T that have helped me in so many different ways. The staff who kept me out of trouble more than once. My fellow graduate students in "McMillin's Group": Tom Roth, Steven Jackson, Li Feng, Anusha Sankara, Aaron Pope, and Michael Catanzaro. The lively discussions were very much appreciated.

I would like to thank my family, especially my parents, grandparents, and sons. Finally I would like to thank my friend, muse, spouse, and guide through the maze of general semantics, Patricia Berens. Our long walks and rambling talks were most helpful. I cannot thank her enough.

# TABLE OF CONTENTS

# LIST OF ILLUSTRATIONS

# LIST OF TABLES

# ACRONYMS

| | |
|---|---|
| **ABS** | Anti-Lock Braking System |
| **ACM** | HRU Access Control Matrix |
| **APT** | Advanced Persistent Threat |
| **BIT** | Belief, Information transfer, and Trust |
| **BLP** | Bell-LaPadula Model |
| **BUT** | Belief, Utterance, and Trust |
| **CPS** | Cyber-Physical Systems |
| **CPS-CSH** | Cyber-Physical Systems in Critical Systems Heuristics |
| **DGI** | Distributed Grid Intelligence |
| **ES** | Event System |
| **FREEDM** | Future Renewable Electric Energy Delivery and Management Systems Center |
| **HIGH** | Higher security partition |
| **IFS** | Information Flow Security |
| **HRU** | Harrison, Ruzzo, and Ullman Model |
| **LTS** | Labeled Transition System |
| **Lipner** | Lipner Model |
| **LOW** | Lower security partition |

| | |
|---|---|
| **MSDND** | Multiple Security Domains Nondeducibility |
| **ND** | Nondeducibility |
| **NF** | Noninference |
| **NI** | Noninterference |
| **PLC** | Programmable Logic Controller |
| **SD** | Security Domain |
| **SSP** | Simple Security Property |
| **SST** | Solid State Transformer |
| ***property** | Star Property |
| **wff** | Well-formed formula |
| **VPN** | Virtual Private Network |

# NOMENCLATURE

| Symbol | Description |
|---|---|
| $\sigma_x$ | state $x$ |
| $s_x$ | boolean state variable, $x$ is true or false |
| $c_x$ | command $x$ |
| $Tr$ | The set of all valid traces over an event system |
| $\tau$ | A specific trace over an event system (ES) |
| $\uparrow$ | Trace restrictor function |
| $\mid$ | Alternate trace restrictor function |
| $\pi T$ | Noniterference purge operator |
| $\bot$ | Falsum or logically *false* |
| $\top$ | Logically *true* |
| $\equiv$ | Tautology, i.e., either side may be substituted for the other |
| $\sim$ | Logical negation: $\top \equiv \sim\bot$ |
| $\wedge$ | Logical AND |
| $\vee$ | Logical OR |
| **xor** | Exclusive OR: $\varphi$ **xor** $\psi \equiv (\varphi \vee \psi) \wedge \sim(\varphi \wedge \psi)$ |
| $\rightarrow$ | Material Implication: $\varphi \rightarrow \psi \equiv \sim\varphi \vee \psi$ |
| $\leftrightarrow$ | Logical Biconditional ("if and only if"): $\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$ |
| iff | Logical Biconditional ("if and only if") |
| $\varphi$ | A boolean statement that can be evaluated in $w \in W$ |
| $\psi$ | A boolean statement that can be evaluated in $w \in W$ |
| $\Phi$ | The set of all validly constructed statements, $\varphi$ |
| $\varnothing$ | The null, or empty, set $\{\}$ |
| $\Rightarrow$ | Logical Implication: $\varphi \Rightarrow \psi \equiv \forall w \in W : \varphi \vDash \psi$ |
| $\Leftrightarrow$ | Logical Bimplication: $\varphi \Leftrightarrow \psi \equiv (\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$ |
| $\Box\varphi$ | The Modal "must be so" operator ("it is always such that…") |
| $\Diamond\varphi$ | The Modal "possible" operator ("it is possible that…") |
| $[i]\,\varphi$ | The Modal "$\varphi$ must be so after $i$ transitions" operator |
| $\langle i\rangle\,\varphi$ | The Modal "$\varphi$ possible after $i$ transitions" operator |
| $w \vdash \varphi$ | The statement $\varphi$ is valid in world $w$ ("yields") |
| $w \vDash \varphi$ | Values from world $w$ cause $\varphi$ to evaluate to TRUE ("models") |
| $\mathfrak{F}$ | A Kripke frame |
| $\mathfrak{M}$ | A Kripke model |
| ${}_wR_{w'}$ | A transition function from world $w$ to $w' : w, w' \in W$ |
| $\mathfrak{R}$ | The set of all transition functions in a complete Kripke Frame |
| $\mathbb{V}_x^i(\varphi)$ | Valuation function of boolean $x$ in domain $i$ |
| $B_i\varphi$ | Modal BELIEF operator |
| $I_{i,j}\varphi$ | Modal INFORMATION TRANSFER operator |
| $U_j\varphi$ | Modal UTTERANCE (broadcast) operator |
| $T_{i,j}$ | Modal TRUST operator |

# 1. INTRODUCTION

This work was inspired by a life-long interest in “locks” and “keys” of various sorts. The problem of keeping the bad guys out is a major part of my career. As security officer for various organizations, I was responsible for electronic security for all of the data centers where I worked. The constant announcements that the security problem was solved *once and for all* by an industry expert’s software re-enforced a deep and abiding skepticism with security as a whole. My experience in my career with security that *almost* worked led eventually to the present work.

The papers that make up the bulk of this dissertation should be viewed in light of the hidden fear inside every security officer that somewhere there is a hole, a gap, an innocent observable result, or that nebulous “something” that gives the adversary the information to take what they will. These papers build towards the final result that, at least in CPS, security does not work in the sense that most people think it does. Perfect security is highly unlikely in CPS and may be cost prohibitive. It may be more realistic to try for the ideal while recognizing the limitations of CPS security. These systems will most likely be breached.

The first paper in this effort is *Modeling and reasoning about the security of drive-by-wire automobile systems* [1]. This paper examines the drive-by-wire automobile attached to a corporate network such as General Motors OnStar. This work leads naturally to *A Multiple Security Domain Model of a Drive-by-Wire System* [2] that introduces Multiple Security Domains Nondeducibility (MSDND)(ES) as a model to handle information flows that cannot be modeled with traditional methods. During the work on these two papers, concerns over applying MSDND(ES) to another model such as Stuxnet and how to describe the roll of trust in these attacks lead to a third paper, *A Modal Model of Stuxnet Attacks on Cyber-Physical Systems: A Matter of Trust.*

Work on the smart grid has pointed out the usefulness of the physical side of the CPS to verify the cyber side and to point out some of the weaknesses of the system, and in the

fourth paper *Breaking Multiple Security Domains Nondeducibility on the Smart Grid* [3]. Security may not work to completely protect a CPS, but we can use security models to find weaknesses. Hopefully, we can then safeguard them and nullify some attacks.

### 1.1. PROBLEM STATEMENT

Security models designed for purely cyber systems or those designed for purely physical systems do not work effectively to discuss CPS. The tools developed in the past are not adequate; newer and more targeted tools must be developed. This work presents MSDND(ES)as new tool to examine CPS. With this new model, it is possible to describe CPS that cannot easily be modeled otherwise.

The simple act of observing the CPS leads to information leaks. CPS of interest normally cannot be hidden nor can the human activities around them. Information leakage via physical observation is subtle and causes unavoidable and unnoticed security issues. A strong case can be made that a CPS cannot be completely secured [4].

One of the major problems with CPS is that the cyber system is tightly coupled, or intertwined, with the physical system. Not only must the cyber system be secured, the physical system must also be protected. Much effort can be placed into securing both systems only to overlook the interface between the two, but these very difficulties can possibly be leveraged to use the physically observable parts of the CPS to verify the cyber parts of the system. Some examples of using the physical side to verify the cyber side will be presented in Sections 6 and 7.

### 1.2. PROBLEMS SPECIFIC TO CYBER-PHYSICAL SYSTEMS

In this work, the problems specific to CPS will be examined. By modeling CPS using the previously available models, the weaknesses and problems with underlying assumptions of these models will become apparent. This does not mean those models are useless when dealing with CPS, far from it. The earlier models are still useful in defining the structure of a CPS and may assist in finding and examining weaknesses in our systems. However, the

existing models do not serve well in terms of actually discovering all threats or in designing systems and procedures to provide adequate security, nor do they provide assistance in explaining why such threats exist. Since absolute security does not exist with CPS; the field must evolve towards resilience in the face of security breaches. Breaches *will* happen.

## 1.3. RELATED WORK

**1.3.1. Historical Security Tools.** In Section 2, some early security models [5] are examined in the light of cyber systems. The HRU model is discussed briefly as a tool to isolate the actors of interest into *subjects* and *objects*. The BLP and Lipner models are presented as a way to begin to structure the security domains and to begin to understand the possible information flows in a particular CPS. In practice, this is a difficult part of the modeling of any CPS. Often these steps are repeated in a process of continual refinement of the model.

**1.3.2. Information Flow Security Models.** Once the subject/object and security domain structure is well understood, the information flow [6] over the Event System (ES) of the model can be mapped. Traditionally, Information Flow Security (IFS) can be examined using a wide range of different models and hybrids of those models. Among the most useful of these for CPS are Noninterference, Noninference [7], and Sutherland's Nondeducibility (ND)(ES) [8]. These will be described in detail in Section 2.

**1.3.3. Sutherland's Nondeducibility (ND).** Introduced by Sutherland in 1986 [8], ND is described in the literature from two different viewpoints which are both explored in this work. The trace based view, see Section 2.3.1 is easier to understand but in some cases less useful than the frame based version. The frame based version introduced by Sutherland is more suited to the analysis of CPS and is discussed briefly in Section 2.3.2. While the concepts are similar, there are striking differences that are sometimes critical to the description of a specific system. Equivalent descriptions of a CPS will be obtained by

using either the trace based or frame based version, but it is not uncommon for one method to be more direct than the other.

### 1.4. METHODOLOGY

The basics of Belief, Information transfer, and Trust (BIT) logic are introduced in Section 3. The most often overlooked issue in research concerning security and CPS is how to deal with the fact that an "agent", human or computer, in some sense must believe and trust other agents [9] [10]. BIT logic is a doxastic modal logic[1], not a propositional logic, and deals with belief and trust in a logical statement $\varphi$ rather than whether a statement is true or false. The BIT logic is strictly concerned with the state of mind of an agent's understanding of the truth value of a statement and must not be confused with the actual truth value of the statement. The BUT logic is essentially the same as the BIT logic, but is concerned with publicly broadcasts of information rather than BIT logic's private transfer from agent $j$ to agent $i$.

Section 3 presents the underlying modal framework for the rest of this work as well as the axiomatic system used, see Table 3.1. Kripke frames [11] [12] and models built upon those frames are key to the understanding of Sutherland's ND [8]. While the explanation is brief, it should provide enough background to grasp the important aspects of ND. Finally, the basic definition of MSDND [2] is presented from a modal viewpoint. Multiple Security Domains ND and the application of BIT Logic to CPS are the heart of this work. MSDND provides a tool to examine real CPS that are usually less than perfectly defined. With this

[1]The BIT operators, $B_i, I_{i,j},$ and $T_{i,j}$ are the doxastic equivalents of the more familiar □ operator.

tool comes the beginnings of a more realistic view of complex security models and CPS and then moves towards a more realistic view of what security can and cannot do.

## 1.5. RESEARCH DIRECTION AND INTERRELATIONSHIPS

Sections 4, 5, 6, and 7 are distilled from published papers. The first paper, *Modeling and Reasoning about the Security of Drive-by-wire Automobile Systems*, raised three questions:

1. What can be done to deal with cases where Sutherland's ND does not apply due to constraints or the lack of valuations in the description of the CPS?
2. How can issues with trust be described precisely?
3. Can the physical side of a CPS be used to verify the cyber side?

To make the evolution of this work easier to follow, the map in Figure 1.1 should be helpful. In short, the questions raised in the first paper, *Modeling and Reasoning about the Security of Drive-by-wire Automobile Systems*[1], related to applying ND to the drive-by-wire car led directly to the development of MSDND in the second paper, *A Multiple Security Domain Model of a Drive-by-Wire System.* MSDND led directly to the Stuxnet paper, *A Modal Model of Stuxnet Attacks on Cyber-Physical Systems: A Matter of Trust* and the introduction of BIT logic provided a way to describe the role of trust in the attack. Corollary 6.6.1 hinted at using the problematic intertwining of the cyber and physical systems as a method to break Nondeducibility and make the attack visible.

All of these questions and new methods were then applied to the electrical smart grid in the final paper, *Physical Attestations, Nondeducibility, and the Smart Grid*, which uses the new MSDND model to describe that CPS.

## 1.6. DRIVE-BY-WIRE AUTOMOBILE

In Section 4, a specific CPS is modeled using the traditional tools [1]. The automobile is modeled in three different modes: Normal operations, hazardous road conditions, and

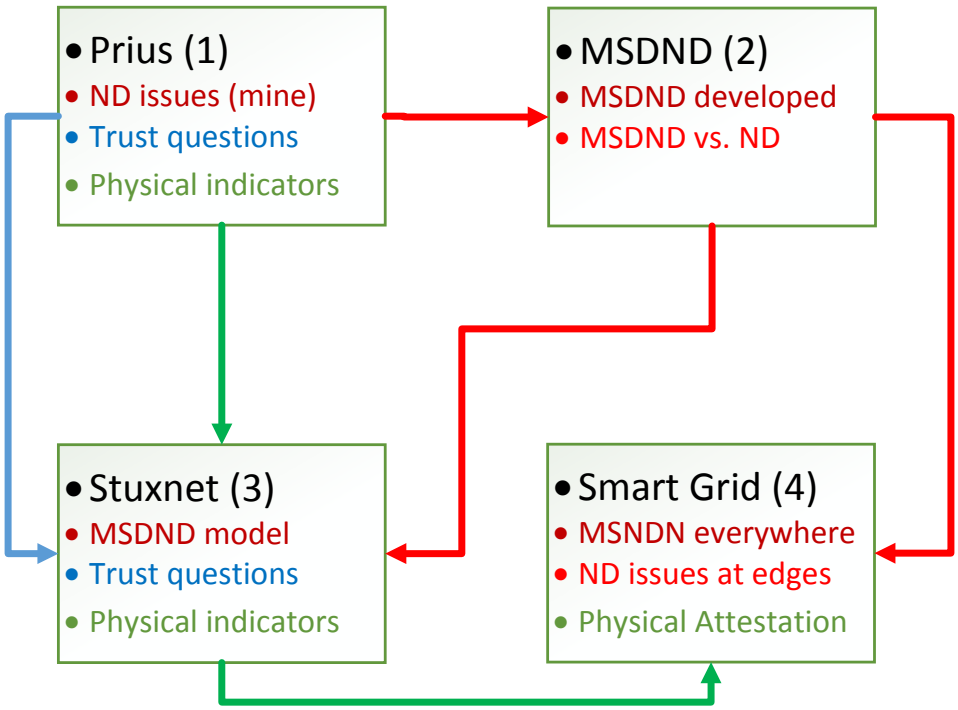



Figure 1.1. Flow of Questions Raised in Drive-by-Wire Paper

remote control via a manufacturer's network connection. The model is built by applying increasingly more sophisticated security models and the strengths and weaknesses of each is presented. The security models presented are HRU, BLP, Lipner Model (Lipner), and three IFS models. The IFS models look at Noninterference, Noninference, and trace-based Sutherland ND. This section not only presents the analysis of a real CPS but also presents the beginnings of an algorithm to analyze a generic CPS.

### 1.7. MULTIPLE SECURITY DOMAINS NONDEDUCIBILITY

MSDND(ES) was developed to address situations where Sutherland's ND applies as well as those where ND should apply but there are problems. MSDND(ES) is defined and

explained fully in Section 5. Some problems that arise when attempting to secure CPS are also explored in Section 5.

### 1.8. STUXNET TYPE ATTACKS ON CPS AND TRUST

In Section 6, MSDND(ES) is used to model [13] the new threat posed by an Advanced Persistent Threat (APT) [14] such as Stuxnet [15]. The goal is not to do another detailed analysis of Stuxnet itself, but rather to try and model such a threat from an IFS viewpoint [16]. In Section 4, it is mentioned that trust plays a key role [17] in attacks on CPS, yet there are few tools to precisely define trust let alone discuss mathematically trust and belief. A new tool, the BIT logic [9] developed by Liau has great promise in this area. BIT logic includes two concepts of a liar which are useful in the analysis of CPS. Many different types of social engineering attacks could be described more clearly, and more compactly, using BIT logic.

### 1.9. MSDND(ES) AND TRUST IN THE SMART GRID

MSDND is used in Section 7 to model the smart grid with Distributed Grid Intelligence (DGI) as envisioned by the Future Renewable Electric Energy Delivery and Management Systems Center (FREEDM) project [18]. One of the more commonly studied attacks on the smart grid is the *fake power injection* attack. The MSDND(ES) nature of the attack is viewed from the cyber messaging viewpoint and the physical measurements of voltage, current, and phase angle on a common distribution bus and is shown to be nondeducible. However, using the approach of physical attestation [19], the dependency of the physical measurements upon known system characteristics and the monitored cyber

messages can be used to break the MSDND(ES) nature of the attack to reveal the malicious node.

## 1.10. DISCUSSION

Some "thought experiments", gedankenversuch[2], in the Section 9 present the ideas of Nondeducibility and information flow security in a clear and easy to understand manner. Some additional topics are briefly discussed to further expand upon the ideas developed during the course of these studies. These include a number of different ways to understand the relationship of MSDND(ES) to other security models along with some of the advantages of using MSDND(ES).

Finally, MSDND is discussed in the light of semantic distance and levels of abstraction. More work needs to be done in these areas, but there is the promise of new uses of MSDND as a method to model more complex information flows across not only security domains but across levels of abstraction such as encountered in recent work with Cyber-Physical Systems in Critical Systems Heuristics (CPS-CSH). In CPS-CSH systems become terribly complex with information flows between levels of abstractions of the system which are very difficult to model. There are many opportunities for further research in the area of CPS-CSH.

[2]Gedankenversuch is German for "thought experiment". Many "experiments" in quantum mechanics can never be done, but important results can still be derived. Probably the most famous gedankenversuch is Schrödinger's Cat.

# 2. RELATED WORK

## 2.1. HISTORICAL MODELS

**2.1.1. HRU.** The HRU [20] is mechanically simple. It consists of a matrix of Subjects, Objects, and Commands [21]. Each element contains the rights for a pair of subjects, objects, or subject and object. This matrix is called the HRU Access Control Matrix (ACM).

The contents of each element of the matrix are manipulated by way of a simple set of commands in either of the two forms:

**Algorithm 1** HRU Conditional Command

**HRU conditional command** $\alpha(X_1, X_2, \ldots, X_k)$
  **if** simple or compound condition **then**
    statement(s)
  **end if**
  end

**Algorithm 2** HRU Unconditional Command

**HRU unconditional command** $\alpha(X_1, X_2, \ldots, X_k)$
  statement
  statement
  statement
  end

In an HRU command, the only allowed statements are: enter, delete, create subject, create object, destroy subject, destroy object. A procedure may consist of multiple commands, but each command has this very simplistic structure.

The goal of the HRU model is to provide a structure that can emulate the desired rights structure and characteristics while still being simple enough to lend itself to mathematical analysis. Usually the goal is to show that all such structures are members of the set of all Turing machines.

While the commands and Access Control Matrix are easy to understand and simple to manipulate, applying the HRU Model to a specific situation is challenging. The real world model may map easily onto the HRU model without being of much direct use in regards to modeling CPS. HRU was intended to be used to model access controls not IFS. However, it is an excellent starting point in understanding the basics of a model of a CPS

**2.1.2. BLP.** A classical BLP model [7] is composed of distinct security levels [22] with each subject or object assigned to an appropriate security level. During the operation of the model, information flows from more secure levels to the less secure levels [23]. Higher levels in the BLP model are said to dominate lower levels, and BLP rights depend solely on this relationship. The BLP model also introduces two new rights concepts, the Simple Security Property (SSP) and the Star Property (*property). The SSP prohibits the reading of any object or subject at a higher level. The *property prohibits the writing of any object or subject at a lower level.

**2.1.3. Lipner Model.** The Lipner model [24] [7] changes the BLP model into a model of trust in the form of integrity levels and integrity compartments which act much like BLP security levels and security domains. The policy of most interest in this dissertation is the ring policy form of the Lipner model, which adds three new rules:

1. Any subject $s \in S$ may read any object $o \in O$ regardless of the integrity levels $i(s), i(o) \in I$.
2. $s \in S$ can write to $o \in O$ if and only if $i(o) \leq i(s)$.
3. $s_1 \in S$ can execute $s_2 \in S$ if and only if $i(s_2) \leq i(s_1)$.

The Lipner model adds the element of trust when used with the BLP model. This allows the system to be described in terms of the integrity or the amount of trust placed

in each of the levels. It should be noted that the integrity levels can be independent of the security levels of the BLP model.

## 2.2. INFORMATION FLOW SECURITY MODELS

**2.2.1. Noninterference.** Noninterference is concerned with actions and events leading to the unintended flow of information from one security domain to another, for example from HIGH to LOW. For the purposes of Noninterference, an action is any act by a subject that causes a change in the rights or states of another subject or object (subjects may initiate actions directly where objects may not). An event is the result of an action where there is an input, an output, a change in the state of the system, or a change in the rights of an object or subject. For example, if a subject grants another subject the rights to read the contents of a write-only object **myfile**, changing the file from "WRITEONLY" to "READ" is an "action". The change in the rights to the object, i.e. the second subject may now read the file when it could not before, is an "event". An action or event does not need to be directly observable in any physical sense, as this example shows. Changing a bit in memory is an action even if we cannot see any change in the physical memory with the naked eye.

Informally, Noninterference (NI) [25] [26] [27] [28] holds when an entity with low security sees the exact same actions or events whether or not high level events are taken into account. In other words, the information seen by the low level entity is the same before and after all high level events are deleted. This can be shown by comparing the trace of events seen by the low level when all actions occur to the trace of events seen by the low level after all high level actions and high level events have been removed from the trace. If the trace seen by the low level entity with all high level actions and events exactly matches the trace seen by the low level entity with all the high level actions and event removed, the system is secure under Noninterference with respect to those actions and events. A formal treatment of Noninterference will now be given.

**Definition 1.** *Security Levels*

With no loss of generality, the system is divided into two security levels. The level examined for noninterference will be denoted by Lower security partition (LOW) and all higher security levels will be denoted by Higher security partition (HIGH). Multiple security levels can be examined pairwise sequentially using this convention.

**Definition 2.** *Commands*

Let the set of all commands be $Z = \{c_0, c_1, \ldots\}$.
A specific sequence of commands is denoted as $c_s = \{c_0, c_1, \ldots, c_n\}$.

**Definition 3.** *Possible states of the system*

The set of all possible states of the system is $\Sigma = \{\sigma_0, \sigma_1, \ldots, \sigma_n\}$ where $\sigma_0$ is some initial state and $\sigma_i$ is the $i^{th}$ state of the system.

**Definition 4.** *Trace* $(T)$

The list of all events seen at a particular security level, e.g., HIGH, is denoted as $T_H$. Likewise, the list of all events seen at a set of multiple security levels is denoted as $T_{2,3}$ where 2 and 3 are the security levels. All traces are ordered sets of events. Also:

$$T(c_0, \sigma_0) = \sigma_1 \tag{2.1}$$

$$T(c_{i+1}, \sigma_{i+1}) = T(c_{i+1}, T(c_1, \sigma_i)) \tag{2.2}$$

.

**Definition 5.** *Outputs*

The set of outputs from a sequence of commands is given by $O = \{o_1, o_2, \ldots, o_n\}$. A specific ordered set of outputs will be denoted as $P^*(c_s, \sigma_i)$.

**Definition 6.** *Purge Operator* $\pi$

The Purge Operator removes all actions by a specific entity. For example, to remove all the actions by subject $A$ the purge function is denoted as $\pi_A$.

**Definition 7.** *Projection Operator Proj*$(\ldots)$

The Projection Operator produces the set of events that a subject or object is allowed to see. This is denoted as $Proj(s, c_s, \sigma_i) \rightarrow P^*(c_s, \sigma_i)$.

**Definition 8.** *Noninterference (NI)*

With respect to a subject, $A$, commands are NI secure with $G'$ users or subjects $(A, G : |G')$ if and only if:

- $\forall c_s \in c^*$ and $s \in G'$
- $proj(s, c_s, \sigma_i) = proj(s, \pi_G(c_s, \sigma_i)$

or

- output$(c_s, G')$ =output(purge$(c_s, G), G')$

**2.2.2. Noninference.** Like Noninterference, the notion of Noninference (NF) [7] is concerned with information flow. Informally, the NF property holds when a low level observer cannot correctly determine if a result of an event is caused by a high level event or a low level event.

**Definition 9.** *Security Levels*

A system can be divided into two security levels without any loss of generality. The level being examined for Noninference (and later for Nondeducibility) is denoted as LOW. All higher security levels are denoted as HIGH. Multiple security levels can be examined one at a time using this convention.

**Definition 10.** *Commands*

The set of all commands is $Z = \{c_0, c_1, \ldots\}$. A specific sequence of commands is given by $c_s = \{c_0, c_1, \ldots, c_n\}$.

**Definition 11.** *Possible system states*

The set of possible states of the system is given by $\Sigma = \{\sigma_0, \sigma_1, \ldots, \sigma_n\}$ where $\sigma_0$ is an initial state and $\sigma_i$ is the $i^{th}$ state of the system.

**Definition 12.** *Event system*

An event system $(ES)$ is the set of all events in the model.

**Definition 13.** *Trace*

The set of valid traces in the system is denoted as $T_r$. A valid trace is denoted as $\tau$.

**Definition 14.** *Trace restrictor*

The trace restrictor operator $\tau \upharpoonright E$ removes all events other than those in the set of events $E$ as seen by a specific observer.

**Definition 15.** *Noninference Secure*

A system is Noninference Secure (NF) if

$$\mathbf{NF}(ES) \equiv \forall \tau \in T_r : (\tau \upharpoonright L) \upharpoonright H = \varnothing \tag{2.3}$$

where $\tau \upharpoonright L$ is a valid trace.

To prove Noninference holds, it is necessary to prove that:

- $\tau \upharpoonright L \in T_r : \tau \upharpoonright L$ is a valid trace in the system
- $(\tau \upharpoonright L) \upharpoonright H = \varnothing$.

The first condition states that if only the low level events in the trace are examined, then the result is a possible set of events that makes sense when applied to the cyber-physical system. The second condition states that if the low level events are examined and the trace is restricted to only high level events, then the resulting trace is empty. There can be no events in the trace that leak information from the high level to the low level. Both conditions must hold in order to prove Noninference.

## 2.3. SUTHERLAND'S NONDEDUCIBILITY (ND)

While Sutherland ND(ES) was originally introduced from a modal viewpoint [8], it is given here from both the modal view and a trace viewpoint [29]. The differences, and the issues arising from those differences, are discussed later in Section 2.3.3.

The property of Nondeducibility(ND) [7] holds if what is observed at a low level is consistent with any number of high level actions. Therefore, no information is leaked to the low level.

Table 2.1. Trace Operators and Terms for ND(ES)

| | |
|---|---|
| $T_L$ | denotes the set of all possible finite sequences of low level events that are legal. |
| $ES$ | denotes the set of possible events. |
| $HI$ | denotes all HIGH-level inputs. |
| $LI$ | denotes all LOW-level inputs. |
| $HO$ | denotes all HIGH-level outputs. |
| $LO$ | denotes all LOW-level outputs. |
| $\|X$ | denotes the trace restrictor which restricts a trace to events at level $X$. |

### 2.3.1. Trace-Based ND(ES).

**Definition 16.** *Nondeducibility*

A system is Nondeducibility secure (ND) if:

$$ND(ES) = \forall \tau_L, \tau_H \in Tr : \exists \tau \in Tr : \tau|L = \tau_L|L \wedge \tau|HI = \tau_H|H. \quad (2.4)$$

**2.3.2. Modal ND(ES).** In the Sutherland model, the valuation functions are the same for all entities in the same security domain. Given a generic case with two valuations, $V_1(w)$ and $V_2(w)$, the definition of Sutherland Nondeducibility [30] with respect to the valuation, $V_2$ for the model is:

$$ND(ES) = (\forall w \in W : V_2(w), V_1(w) \neq \varnothing)(\exists w^{'} : [V_1(w) = V_1(w^{'})] \wedge [V_2(w) = V_2(w^{'})]). \quad (2.5)$$

If the two domains are not equal, as in *HIGH* and *LOW*, a slightly modified version of equation 2.5 may be used:

$$ND(ES) = (\forall z \neq \varnothing : V_1^{-1}(z) = w \in W)(\exists w^{'} \in W : [V_1(w') = z] \wedge [V_2(w) = V_2(w^{'})]). \quad (2.6)$$

**2.3.3. Trace Based Verses Modal Frame Based Sutherland Nondeducibility.** There is an important difference between the Sutherland Nondeducibility from a trace analysis and the more useful modal frame viewpoint. In a frame, such as defined earlier in Section 3.2, time can be thought of as simply one of the state variables changing in an ordered manner. The state of ND(ES) can be evaluated via the frame based model as the system progresses. A modal frame model is not reduced to analyzing the past behavior of the system or rerunning the system to compare traces. Instead it is possible to analyze the state of the system at any time. Indeed, time in the traditional sense is not an issue. It is important to understand that the final results of trace based or frame based ND(ES) *will be the same.* A set of events in a system are ND(ES) or they are not. Indeed, some systems will lend themselves to trace based analysis more than frame based.

# 3. METHODOLOGY

## 3.1. SECURITY ALGORITHM

Developing a security model is part methodology and part art. The problem lies in the fact that information is always viewed in light of an existing mind frame. The internal mental framework of the analyst can cloud the details and shortcomings of a security model. The art lies in catching these internal blind spots before they can be exploited by an adversary.

To aid in the creation of a model of a CPS, an algorithm is presented. These algorithms are designed to lead a team of analysts through the process. At first glance, the algorithms appear to endlessly circle around the problem without making much progress. Unfortunately, this is a possible outcome of a team analysis. If done properly, the process refines the model over and over until a strong model is developed.

The first algorithm, Algorithm 3 CONTINUALREFINEMENT, is an overview of the basic procedure to build and refine a model. In reality, the process is one of continual refinement and usually will involve retracing steps already completed. This should not be discouraged, but it is entirely possible a team could get caught in the details and lose sight of the eventual goal of developing a CPS model. Hopefully, the algorithm will lead to a spiraling in on the final model instead of a constant circling around the same steps.

The second algorithm, Algorithm 4 CYBERSECURITY, provides a method to build the cyber side of the model using traditional security methods. While this dissertation has shown such a model is not complete for a CPS, it is a necessary first step. Building a cyber side model is essential to determining what entities are important to the model. At each step refinements to the model may require the team or analyst to retrace their work back to the HRU question of what are the subject and object of the model. As the model is refined, these return steps should become a matter of checking to insure the refinements have not

created new issues for HRU or the other traditional models. At the end of each return to the algorithm, the result should be a well defined model of the cyber security side of the CPS.

The next algorithm, Algorithm 5 PHYSICALSECURITY, depends highly on the actual CPS. Some systems will be self-contained or within a highly controlled environment, but most CPS are distributed over a significant distance. With such systems it is critical to include in the models any communications system such as Virtual Private Network (VPN) built over the INTERNET. Physical security must take into account the fact that all CPS can be observed to some extent. Fortunately, physical security and physical attacks are relatively well understood.

**Algorithm 3** CPS Security Methodology

```
 1: procedure CONTINUALREFINEMENT
 2:     refinements = true
 3:     while refinements do
 4:         refinements:=false
 5:         CYBERSECURITY(refinements)
 6:         PHYSICALSECURITY(refinements)
 7:         CYBERPHYSICALINTERFACESECURITY(refinements)
 8:         TRUSTSECURITY(refinements)
 9:         BUILDMODALMODEL(refinements)
10:         KNOWNATTACK(refinements)
11:     end while
12: end procedure
```

**Algorithm 4** Cyber System Security

```
1: procedure CYBERSECURITY(refinements)
2:     procedure HRU MODEL(actors)
3:         Identify and refine subjects and objects
4:         Build or refine possible event system (ES)
5:         Identify security domains SD_i
6:
Ensure: ⋃_i^n SD_i = security universe
7:         procedure BLP MODEL(traditional Security Domains)
8:             Identify and refine security domains
9:             if any subject or object changed then
10:                refinements:=true
11:                restart HRU Model
12:            end if
13:            procedure LIPNER(Trust Compartments)
14:                Identify and refine trust compartments
15:                if any subject or object changed then
16:                    refinements:=true
17:                    restart HRU Model
18:                end if
19:                if any security departments changed then
20:                    refinements:=true
21:                    restart BLP Model
22:                end if
23:            end procedure
24:        end procedure
25:    end procedure
26: end procedure
```

**Algorithm 5** Physical Security

```
1: procedure PHYSICALSECURITY(refinements)
2:     Identify and refine physical security issues
3:     Identify and refine opportunities for physical monitors
4:     Identify and refine opportunities for physical attestation
5:     if changes then
6:         refinements:=true                ▷ Back up to HRU and refine model
7:     end if
8: end procedure
```

**Algorithm 6** Cyber-Physical Interface Security

```
1: procedure CYBER-PHYSICALINTERFACESECURITY(refinements)
2:     Identify existing physical monitors
3:     Identify useful observations[3]
4:     Identify and refine opportunities for physical attestation
5:     if changes then
6:         refinements:=true                ▷ Back up to HRU and refine model again
7:     end if
8: end procedure
```

**Algorithm 7** Trust Security

```
1: procedure TRUSTSECURITY(refinements)
2:     Identify and refine agents                ▷ subject are often agents
3:     Examine model for physical attestations
4:     Examine model for cyber attestations
5:     if changes then                           ▷ Back up to HRU and refine model
6:         refinements:=true                     ▷ Back up to HRU and refine model
7:     end if
8: end procedure
```

**Algorithm 8** Build a Modal Model

```
procedure BUILDMODALMODEL(refinements)
    Determine which questions, φ, are relevant
    Develop valuations for φ
    if ∄ 𝕍_φ^SD(w) then
        Try to develop MSDND(ES)
    end if
    if any progress towards model then
        refinements:=true
    end if
end procedure
```

**Algorithm 9** Explore Known Attacks on Similar CPS

```
procedure KNOWNATTACKS(refinements)
    Literature search                                  ▷ Look for attack on similar CPS
    if new attack found then
        Review results of CyberSecurity
        Review physical attestation
        if cbyer security changes then
            refinements:=true
            restart Cyber Security
        else if physical security changes then
            refinements:=true
            restart Physical Security
        else if interface changes then
            refinements:=true
            restart Cyber-Physical Interface Concerns
        else
            refinements:=true
            restart Trust Security
        end if
    end if
end procedure
```

## 3.2. MODAL FRAMES AND LOGIC SYSTEM

**3.2.1. Modal Logic Models over Frames.** The modal models in this dissertation are built over generalized Kripke frames [11] [33] [12] [34] and require at least some background in modal logic. This section will present an overview of Kripke frames and the axiomatic system used throughout this dissertation.

The set of worlds, $\{w_i \in W : i = 0, 1, \ldots, n\}$ can be thought of as all possible combinations of $m$ boolean state variables $s_0, s_1, \ldots, s_m$.

The possible worlds are connected by a *completely populated* set of transitions, $\{wRw'\}$, where $R{+}_{s_x}$ sets the state variable, $s_x = \top$("true") and similarly, $R{-}_{s_x}$ sets the state variable, $s_x = \bot$("false"). For example, the change of a state variable $s_x$ leads to an irreducible 'jump' from world $w_{s_{x+}}$ to world $w_{s_{x-}}$ [35]. This defines the effect of changing a state variable, e.g., $R{+}_{s_i}$, as a transition from the current world, $w$, to another world,

$w'$ where all other state variables retain their current values and $s_i = \top$ regardless of the previous value of $s_i$. $R{-}_{s_x}$ is viewed in a similar fashion as setting the state variable, $s_x$ to "false" and leaving all other variables unchanged in the transition, $wRw'$, from world $w$ to $w'$. This could result in a transition from one world, $w$, back to the same world, i.e. $wRw$[4]. Transitions of the type $wRw' = \varnothing$ can be disallowed without any loss of generality[5]. This simply states that transitions must move from one world in the frame to another world in the frame, i.e., state changes cannot transition out of the set of all possible worlds $W$. Together, the set of worlds and transitions define a frame, $\mathfrak{F} = \{W, \mathfrak{R}\}$ [36].

At this point it is helpful to informally define some modal logic symbols. The terms "models" and "satisfied" in this usage is the same as in SATISFIES problems.

| | |
|---|---|
| $\Box\varphi$ | $\varphi$ is always true here |
| $\Diamond\varphi$ | $\varphi$ **might** be true here |
| $w \vdash \varphi$ | in $w, \varphi$ is the conclusion of a valid proof |
| $w \vDash \varphi$ | in $w$, states are such that $\varphi$ is true, i.e. $\varphi$ is "satisfied" in world $w$ |
| BIT | The BIT/BUT operators are explained in Section 3.4 |
| $B_i\varphi$ | Belief is a doxastic version of the $\Box\varphi$ operator. |
| $I_{i,j}\varphi$ | Information transfer is a doxastic version of the $\Box\varphi$ operator. |
| $U_j\varphi$ | Utterance is a doxastic version of the $\Box\varphi$ operator. |
| $T_{i,j}\varphi$ | Trust is a doxastic version of the $\Box\varphi$ operator. |

Because $\Box$ and $\Diamond$ are duals, only one need be defined and the other can be used as simply a shorthand. In this dissertation, $\Diamond\varphi$ will be defined as $\exists w \in W : w \vDash \varphi$. $\Box\varphi$ then becomes a shorthand, i.e. $\Box\varphi \equiv \sim \Diamond \sim\varphi$. This translates loosely to: "it is such that $\varphi$ is true everywhere" can be substituted for "it is not possible for $\varphi$ to be false somewhere."

The set $\varphi, \psi \in \Phi_0$ is a set of countably many atomic propositions. Questions can be asked by evaluating well-formed formulas(wff's) built from these atomic propositions. The

[4]This is referred to as a Reflexive Frame.
[5]The frame is a Serial Frame.

set of well-formed formulas is the least set containing $\Phi_0$ that is closed under the following formulation rules:

- if $\varphi$ is a wff, so are $\sim\varphi, \Box\varphi, ,$ and $\Diamond\varphi$
- if $\varphi$ is a wff, so are $B_i\varphi$, and $\sim B_i\varphi$,
- if $\varphi$ is a wff, so are $I_{i,j}\varphi$, and $\sim I_{i,j}\varphi$,
- if $\varphi$ is a wff, so are $T_{i,j}\varphi$, and $\sim T_{i,j}\varphi$,
- if $\varphi$ and $\psi$ are wff, then so is $\varphi \vee \psi$
- if $\varphi$ and $\psi$ are wff, then so is $\varphi \wedge \psi$

As usual, other classical logical operators $\wedge$ (and), $\rightarrow$ (material implication), **xor** (exclusive OR) and $\leftrightarrow$ (if and only if), can be defined as abbreviations. The modal operator, $\Box\varphi$, is an abbreviation for $\sim \Diamond \sim\varphi$ [12] [11] [37]. Because the modal operators $\Box$ and $\Diamond$ are duals, only one needs to be axiomatically defined. In this dissertation, the $\Diamond\varphi$ operator will be the fundamental operator. The axiomatic system is given in Table 3.1. The "K" and "M" axioms are only required to insure that our axiomatic system is correct and complete in order to correctly claim the set $\varphi, \psi \in \Phi$ is closed over the normal propositional operators.

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

### 3.3. TWO MODAL LOGIC BASED MODELS

The classical Sutherland Nondeducibility Model and the Multiple Security Domain Nondeducibility Model are modal models over frames. Informally, each possible combination of binary state variables defines a world, $w$. Changes in any state variable cause a transition to a different world, much like a labeled transition state machine. A framework of the possible combinations of states, the transitions between those combinations, and the

valuation functions for all the states that make up a world, defines the actions of a model. If built properly, this model should behave in the same manner as the CPS. A series of worlds and the translations between those worlds makes up an event system, ES.

**3.3.1. Modal Logic Model.** Recall that the combination of states, $s_0, s_1, \ldots, s_m$ are such that each combination corresponds to a single unique world, $w \in W$. A change in an state variable moves to a different world, $w' \in W$ where all other state variables retain their values. This transition is unique and is denoted as $wRw' \in \mathfrak{R}$. Together, the set of worlds and transitions define a frame, $\mathfrak{F} = \{W, \mathfrak{R}\}$.

The set of all valuation functions, $\{V\}$, is such that $\mathbb{V}^i_{s_x}(w)$ 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. A model is defined as a tuple $M = \{\mathfrak{F}, V\}$ or $M = \{W, \mathfrak{R}, V\}$.

**3.3.2. Sutherland Nondeducibility Model.** In the Sutherland model, the valuation functions are the same for all entities in the same security domain. Typically, some evaluations are restricted to one domain while others appear to span multiple domains. Consider a generic case with two valuations, $\mathbb{V}_1(w)$ and $\mathbb{V}_2(w)$. Sutherland Nondeducibility [30] with respect to the valuation, $\mathbb{V}_2$ for this model is:

$$\begin{aligned} ND(ES) = \quad & (\forall w \in W : \mathbb{V}_2(w), \mathbb{V}_1(w) \neq \varnothing) \\ & \exists w' : [\mathbb{V}_1(w) = \mathbb{V}_1(w')] \wedge [\mathbb{V}_2(w) = \mathbb{V}_2(w')] \end{aligned}$$

**3.3.3. Multiple Security Domains Nondeducibility Model.** Computer security tools work relatively well for computers, but CPS leak information because the physical part 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.

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, see Fig. 3.1. 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 flows, the security domain picture gets complicated, see Fig. 3.2. We need new models that can model the cyber and physical components of CPS.

If the system consists of more than two security levels, the assumption is that the progressively higher security levels are contained within the next lower security level. Indeed, this model has served well since the advent of the Medieval castle with its outer walls and interior keep, see Figure 3.1. Unfortunately, this model does not work well for modern CPS because the security levels often overlap and are rarely entirely contained as in the case of the Medieval castle and keep system, see Figure 3.2.

Good security models exist for computer systems and access control, but CPS are more complicated than the computer systems these models were designed to describe. Information flow security is important and in CPS it is not possible to limit the goal to simply denying information flows from one security domain to another. The physical nature of the system leaks information and novel attacks, such as Stuxnet [15] have shown the limitations of viewing threats as only attempts to steal information. CPS information flow models must now explore information flow across security domain boundaries as being critical to the safe operation of the system.

Extending existing models to multiple security domains is problematic. This dissertation offers a different approach, the Multiple Security Domains Nondeducibility Model. Assume an entity $i$ as any part of the system capable of independent observation or action. The Event System (ES) divides into multiple security domains, $SD^i$, as viewed by each entity $i$ in the model. These domains may, or may not, overlap. These multiple security domains conform to the following rules:

$$\bigcup_{i \in I} SD^i = (ES). \tag{3.1}$$

$$\forall i, j : 0 < i \neq j \leq n : \left[ (SD^i \cap SD^j = \varnothing) \vee (SD^i \cap SD^j \neq \varnothing) \right] \tag{3.2}$$

.

The first rule states that every event happens in some security domain. Unless specifically noted, the event system can be thought of as the universe with security domains being a number of subsets of the universe. Any event that appears to happen outside of the security domains under examination, explicitly happens in that part of the universe that is the unknown domain. This unknown domain is those unexamined surroundings in which our model is defined. For example, an unsuspected adversary in the hills looking down into the parking lot is in the unknown domain.

The second rule states explicitly that any pair of security domains may overlap, may be disjoint, or one may be a contained subset of the other. Other IFS models require disjoint security domains that are separated by an ideal barrier across which no information may flow in *either* direction [7]. MSDND(ES) simply requires the security domains be defined.

**Definition 17.** *Multiple Security Domains Nondeducibility*

There exists some world with a pair of states where one must be true and the other false (exclusive OR), but an entity $i$ has no valuation function for those states. In security domain $SD^i$, $i$ simply cannot know which state is true and which is false. MSDND over an ES can be defined as follows:

$$\text{MSDND(ES)} = \exists w \in W : w \vdash \Box\left[(s_x \vee s_y) \wedge \sim(s_x \wedge s_y)\right] \wedge \left[w \vDash (\nexists\, \mathbb{V}_x^i(w) \wedge \nexists\, \mathbb{V}_y^i(w))\right]. \tag{3.3}$$

An equivalent formulation would be:

$$\text{MSDND(ES)} = \exists w \in W : w \vdash \Box\left[s_x \textbf{ xor } s_y\right] \wedge \left[w \vDash (\nexists\, \mathbb{V}_x^i(w) \wedge \nexists\, \mathbb{V}_y^i(w))\right]. \tag{3.4}$$

In the special case where $s_x$ is $\varphi = \top$ and $s_y$ is $\sim\varphi = \top$, MSDND(ES) reduces to:

$$MSDND(ES) = \exists w \in W : w \vdash \Box\left[\varphi \textbf{ xor } \sim\varphi\right] \wedge \left[w \vDash (\nexists\, \mathbb{V}_\varphi^i(w))\right]. \tag{3.5}$$

### 3.3.4. Reduction of the Sutherland Model to MSDND.

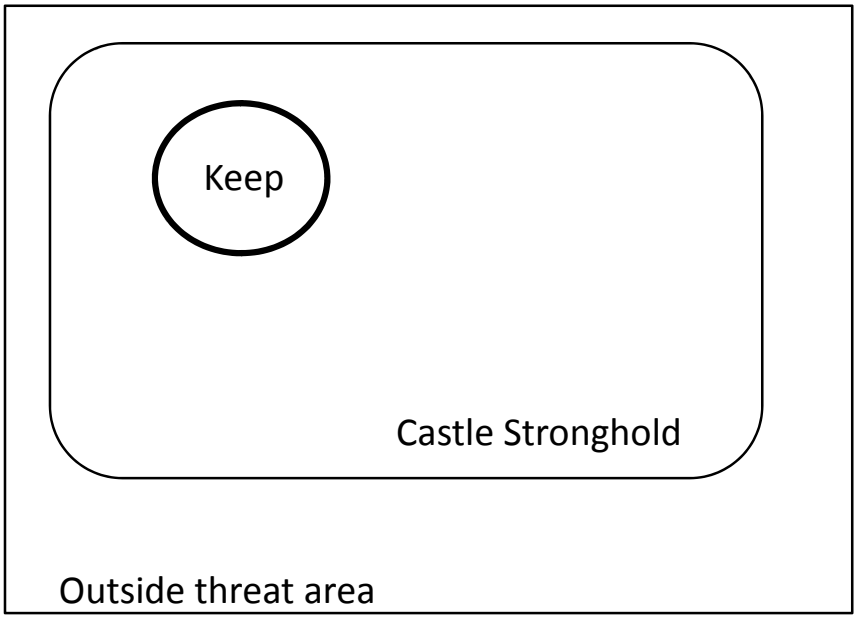



Figure 3.1. A Medieval Castle Model of Security

**Theorem 1.** *Any arbitrary case where ND(ES) holds can be shown to be a special case of MSDND(ES).*

*Proof.*

Given: A system with two security domains, *left* and *right*, and two distinct worlds $w', w'' \in W$ where $ND(ES)$ holds. NOTE: The use of *left* and *right* as designations is to emphasize that MSDND is not a high/low hierarchy model, but is instead a partitioning model.

With no loss of generality, this valuation can be expressed as a binary decision value because in the current world, $w$, either the *right* event has occurred ($w'$) or it has not ($w''$). Create two state variables such that $st \rightarrow (w = w')$ and $sf \rightarrow (w = w'')$. Because this case is $ND(ES)$, it follows that the *left* domain cannot evaluate either $st$ or $sf$ because to do so would break $ND(ES)$. It is now easy to construct the conditions for MSDND(ES).

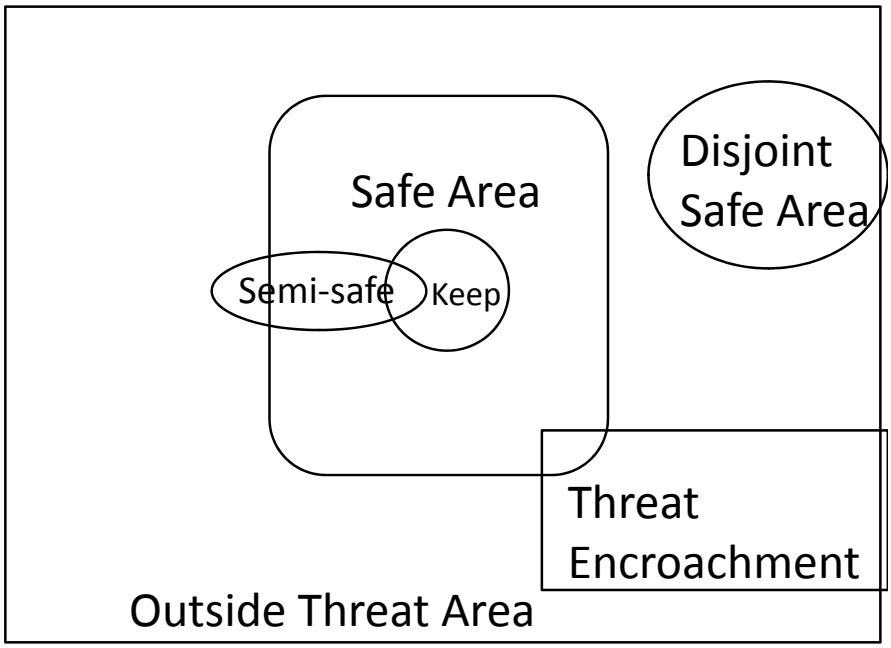



Figure 3.2. Problematic Overlapping Security Domains

| | |
|---|---|
| $w', w'' \in W \rightarrow w \in W$ | By construction |
| $w \vdash \Box(st \textbf{ xor } sf)$ | By construction |
| $\nVdash \mathbb{V}_{st}^{left}(w)$ | ND(ES) |
| $\nVdash \mathbb{V}_{sf}^{left}(w)$ | ND(ES) |
| $w \vDash \left[\nVdash \mathbb{V}_{st}^{left}(w) \wedge \nVdash \mathbb{V}_{sf}^{left}(w)\right]$ | ND(ES) |

Since the first clause is constructed as a tautology, by *conjunction* we can construct the conditions for MSDND(ES).

$$\text{MSDND(ES)} = \exists w \in W : w'' \vdash \Box(st \textbf{ xor } sf) w \vDash (\nVdash \mathbb{V}_{st}^{left}(w) \wedge \nVdash \mathbb{V}_{sf}^{left}(w)). \qquad (3.6)$$

□

**3.3.5. Remarks about the Reduction from Sutherland Model to Multiple Security Domains Model.** It is possible to reduce any system that meets ND(ES) to one that meets MSDND(ES) by defining decision variables for each variable that is ND(ES). However, a reduction in the other direction, MSDND(ES) to ND(ES) is not always possible. MSDND(ES) works even in the case where the model under examination does not contain a valuation function capable of returning the value of $\varphi$ in all worlds. Sutherland's ND(ES) does not address this situation.

Furthermore, MSDND(ES) does not depend upon examining two domains nor upon any relationship between those domains such as *low* and *right*. The domains in question might be one wholly contained in the other, they might overlap, or they might be disjoint. There is no need to determine in advance what the relationship is between the two, three, or more domains in question.

## 3.4. DOXASTIC BIT AND BUT LOGIC

Recall from the earlier discussion of the axiomatic system, see Table 3.1, the two generic[6] modal logic operators $\Box\varphi$ (it must always be true that $\varphi$) and $\Diamond\varphi$ (it might be

[6]Here generic modal logic is a more familiar term for what is properly termed alethic logic.

true under some conditions that $\varphi$). These generic modal logic operators often take on more specific meanings for more specific logics such as temporal logic, deontic logic, epistemology, and others.

Because $\Box$ and $\Diamond$ are duals, only one need be defined and the other can be used as simply a shorthand. In this dissertation, $\Diamond\varphi$ will be defined as $\exists w \in W : w \Vdash \varphi$. $\Box\varphi$ then becomes a shorthand, i.e., $\Box\varphi \equiv \sim \Diamond \sim\varphi$. This translates loosely to: "it is such that $\varphi$ is true everywhere" can be substituted for "it is not possible for $\varphi$ to be false somewhere."

BIT and Belief, Utterance, and Trust (BUT) Logic[7] were introduced by Liau [9] [40] to provide a modality to formally reason about belief, information transfer, utterances, and trust when dealing with cyber entities. While it was developed primarily for handling trust in database and distributed systems, BIT logic is useful for describing CPS, especially when humans are involved. Before BIT logic, social engineering attacks could only be described by a narrative in imprecise language. With BIT logic, spoofing and other unwanted behavior is described with simple, formal proofs.

BIT logic is designed to reason about the belief and trust an entity $i$ has in information from an entity $j$, e.g., the belief and trust an operator has in the reading from a monitoring station. The doxastic modal operators that correspond to the usual $\Box$ operator[8] are:

- $T_{i,j}\varphi$ defines the trust $i$ has in a report from $j$ that $\varphi$ is true
- $B_i\varphi$ defines the belief by $i$ that $\varphi$ is true; it does not matter if $\varphi$ is true or not, $i$ believes it to be true
- $I_{i,j}\varphi$ defines the transfer of information directly from one agent to another[9], that is $j$ reported to $i$ that $\varphi$ is true
- $U_j\varphi$ defines the broadcast of information that $\varphi$ is true by an agent $j$. No efforts are made to hide the transmission although the actual message may be obscured.

[7] We will not distinguish between BIT and BUT logic as it will be obvious which we are using.
[8] For our purposes, we do not need the doxastic versions of $\Diamond$.
[9] If the information is published to the world at large, the Utterance operator, $U_j\varphi$, is used.

BIT Logic introduces four new modal operators to deal with what agents believe, trust, and communicate with each other. Great care must be taken to distinguish between what is true, what is believed to be true, and what reported information is believed and trusted.

**3.4.1. Belief.** It is critical to remember that *belief* in the context of doxastic logic has nothing to do with the notion of the *truth* of a statement $\varphi$. An entity *believes* the truth of a statement based upon some internal state, not based upon any demonstration or proof of the actual statement $\varphi$. In the context of BIT logic, the belief operator, $B_i\varphi$, is a variant of the more familiar modal operator $\Box$. For simplicity, belief is always absolute and there is no corresponding operator for the modal $\Diamond$ nor does the current work require one.

There is an important distinction between $\sim B_i\varphi$ ($i$ does not believe $\varphi$ to be true) and $B_i\sim\varphi$ ($i$ believes $\sim\varphi$ to be true). In the first case, $i$ does not believe in the truth of $\varphi$ while in the second case $i$ is certain that $\varphi$ is false. Again, nothing is said or known about the truth of $\varphi$, but $i$ believes in its own knowledge of the state of $\varphi$.

**3.4.2. Information Transfer.** The information transfer operator, $I_{i,j}\varphi$, clearly states how $i$ gains knowledge of $\varphi$. The information transfer operator inherently assumes $j$ will not lie to $i$; however, this restriction allows liars to lie to trusting agents. It is key to realize the information is transferred directly to an agent who has no direct way to evaluate whether or not $j$ is a liar. Whether or not $i$ thinks $j$ is a liar is determined by the Trust operator, but either way the information is transferred.

There is a difference between the two statements $I_{i,j}\sim\varphi$ (read "$j$ told $i$ that $\varphi$ is not true") and $\sim I_{i,j}\varphi$ (read "$j$ did not tell $i$ anything about $\varphi$). In this case the difference is obvious, but care must be taken in how information transfer statements are read and used.

**3.4.3. Utterances.** If information is not transferred directly from agent $j$ to $i$ but is instead broadcast to any and all agents, the Utterance operator $U_j$ is used to represent the transfer. For example, if agent $j$ sends up a flare, that information $\varphi$ can be seen by everyone and is no longer private. The Information transfer operator, $I_{i,j}\varphi$, is no longer appropriate and the Utterance operator, $U_j\varphi$, is used instead.

The statement $U_j\varphi$ is interpreted as agent $j$ openly puts the information $\varphi$ out for any entity to know. Any agent $i$ has no direct way to evaluate whether or not $j$ is a liar. Whether or not $i$ thinks $j$ is a liar is determined by the Trust operator, but either way the information is transferred.

The statements $U_j{\sim}\varphi$ (read "$j$ openly said that $\varphi$ is not true") and ${\sim}U_j\varphi$ (read "$j$ did not say anything about $\varphi$) are not the same concept. The differences between these two statements is easy to understand, but care must be taken not to confuse them.

**3.4.4. Trust.** The trust operator, $T_{i,j}\varphi$, is a doxastic modal operator for trust that corresponds to the □ modal operator and is used to describe the internal state of trust that $i$ has for knowledge about the state of $\varphi$ learned directly from $j$. Two subtle distinctions are important here. First, the trust that $i$ has in $j$ about $\varphi$ has *no bearing* on the trust $i$ has in $j$ about the state of any other information $\psi$. Second, trust in a report does not imply the information has ever been transferred, only that it will be trusted once $j$ actual makes a report or transfers the information about $\varphi$. For the purposes of this dissertation, $T_{i,j}\varphi \rightarrow T_{i,j}{\sim}\varphi$; i.e., if $i$ trusts a positive report it is assumed $i$ will trust a negative report as well.

The two statements, $T_{i,j}{\sim}\varphi$ (read "$i$ trusts any report from $j$ that $\varphi$ is not true") and ${\sim}T_{i,j}\varphi$ (read "$i$ emphatically does not trust any report from $j$ about $\varphi$ at all"), are not equivalent in any circumstances. They are contradictory.

Table 3.1. The Axiomatic System

1. Definition of logical and modal operators (abbreviations)
   - D1: $\varphi \wedge \psi \equiv \sim(\sim\varphi \vee \sim\psi)$
   - D2: $\varphi$ **xor** $\psi \equiv (\varphi \vee \psi) \wedge \sim(\varphi \wedge \psi)$**(Exclusive OR)**
   - D3: $\varphi \rightarrow \psi \equiv \sim\varphi \vee \psi$
   - D3: $\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$
   - D4: $\Diamond\varphi \equiv \exists w \in W : w \vDash \varphi$
   - D5: $\Box\varphi \equiv \sim \Diamond \sim\varphi$
   - D6: $B_i(\varphi)$ Entity $i$ believes the truth of $\varphi$
   - D7: $I_{i,j}(\varphi)$ Entity $j$ informs $i$ that $\varphi \equiv \top$
   - D8: $T_{i,j}\varphi$ Entity $i$ trusts the report from $j$ about $\varphi$

2. Axioms
   - P: all the tautologies from the propositional calculus
   - **K**: $\Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)$
   - **M**: $\Box\varphi \rightarrow \varphi$
   - S4: $\Box\varphi \rightarrow \Box\,\Box\,\varphi$
   - S5: $\Diamond\varphi \rightarrow \Box \Diamond \varphi$
   - A1: $\sim \Box\, \varphi \rightarrow \Box\sim \Box\, \varphi$
   - A2: $\Diamond(\varphi \vee \psi) \rightarrow \Diamond\varphi \vee \Diamond\psi$
   - A3: $\Box\varphi \wedge \Box\psi \rightarrow \Box(\varphi \wedge \psi)$
   - B1: $[B_i\varphi \wedge B_i(\varphi \rightarrow \psi)] \rightarrow B_i\psi$
   - B2: $\sim B_i\bot$
   - B3: $B_i\varphi \rightarrow B_iB_i\varphi$
   - B4: $\sim B_i\varphi \rightarrow B_i\sim B_i\varphi$
   - I1: $[I_{i,j}\varphi \wedge I_{i,j}(\varphi \rightarrow \psi)] \rightarrow I_{i,j}\psi$
   - I2: $\sim I_{i,j}\bot$
   - C1: $B_iI_{i,j}\varphi \wedge T_{i,j}\varphi \rightarrow B_i\varphi$
   - C2: $T_{i,j}\varphi \equiv B_iT_{i,j}\varphi$

3. Rules of Inference [38] [11]
   - R1: from $\vdash \varphi$ and $\vdash \varphi \rightarrow \psi$ infer $\psi$ **(Modus Ponens)**
   - R2: $\sim(\varphi \wedge \psi) \equiv (\sim\varphi \vee \sim\psi)$ **(DeMorgan's)**
   - R3: from $\vdash \varphi$ infer $\vdash \Box\varphi$ **(Generalization)**
   - R4: from $\vdash \varphi \equiv \psi$ infer $\vdash \Box\varphi \equiv \Box\psi$
   - R5: from $\vdash \varphi \equiv \psi$ infer $\vdash T_{i,j}\varphi \equiv T_{i,j}\psi$

Note: Axioms **K** and **M** are required to insure correctness and completeness of the logical system [39].

# 4. DRIVE-BY-WIRE AUTOMOBILE

In this section we will examine the interactions of a network connected drive-by-wire automobile with a "trusted"[10] network for roadside assistance such as General Motor's OnStar.

An increasing number of modern automobiles are essentially drive-by-wire systems, highly computerized, and connected wirelessly to services such as OnStar or Toyota Safety Connect. While these features enhance automobile safety and reliability, the security impact is a growing concern. This section examines the security of drive-by-wire automobile systems. Generic models of access control and information flow are defined, with specific instances of the 2010 Toyota Prius used where appropriate. The automobile systems are examined from the viewpoint of the driver with special emphasis on the driver's ability to determine who, or what, is actually in control of the automobile in critical situations.

## 4.1. INTRODUCTION

Modern automobiles are essentially drive-by-wire systems connected wirelessly to manufacturers networks. The security impact of these advanced features is a growing concern. Three fundamental questions are: (i) what is an appropriate security model for such systems? (ii) what level of security do such systems provide? and (iii) how does the driver interact with the automobile and manufacturer? To address these questions, this section models automobile systems as CPS and applies classical security models of multilevel access control and information flow security. Compared with information and computer systems, CPS present an interesting perspective of security from three standpoints. First, CPS have at their heart a control mechanism that is either a computer system or something similar; this raises the issue of securing a computer system. Second, the physical side of a cyber-physical system makes it easy to understand how merely observing the system can

[10] Here trusted is used in its more general sense and not in the BIT Logic sense.

leak information; this eases the difficulties of visualizing attacks on the system. Third, CPS are prone to collateral damage when compromised. It is much easier to see the implications of a breach when it leads to total destruction of the system rather than simply accessing a few confidential documents in the case of an information security breach. There are two important questions to answer. First, why should one analyze how a drive-by-wire automobile interacts with a remote roadside assistance service offered by the manufacturer? The answer is that the presence of a drive-by-wire system means that the driver has limited control of the automobile in certain situations. Indeed, in light of malware such as Stuxnet [15], an attack on this CPS can have life-threatening consequences. Second, how should the analysis be done? Why not simply discuss anecdotal evidence? The answer is that a framework is needed if the system is to be understood objectively. The formal security models studied in this dissertation support the analysis of the interactions between the driver, drive-by-wire automobile and remote assistance services. This section specifically considers the Toyota Safety Connect system, which has many of the same features as the OnStar system. A prime concern is whether or not the driver can determine if the automobile is under his/her control, the control of the on-board computer, or under the control of an external entity. If the driver is not in control, is there anything he or she can safely do? Or is the driver helpless? We will show that, in some situations, the driver not only has no control, but that the driver cannot determine who or what has control of the automobile. More than most current automobile models, the Toyota Prius is a drive-by- wire system. In fact, other than issuing various requests to the controller area network (CAN), which is the control mechanism of the Prius, the driver has little to do as far as our security model is concerned. For example, when the driver steps on the brake pedal, a sensor is activated that requests the Prius to engage the braking system. There is no mechanical linkage and no possibility of the driver overriding an errant command or forcing the Prius to obey some other command. Indeed, the Prius traction control system takes full control of the accelerator, braking and all other systems from the driver under hazardous road conditions. Should the Prius determine not to honor a command, the driver has no more control than

a passenger. This dissertation considers several security models and analyzes three issues for each model: (i) normal operations that take the automobile from a powered down to a driving state; (ii) traction control under hazardous road conditions; and (iii) external (remote) control of the automobile. Special emphasis is placed on the driver's ability to determine who, or what, is actually in control of the automobile.

## 4.2. FUNCTIONAL MODEL

The functional model consists of a corporation that provides services to its automobiles in the form of remote assistance (e.g., navigation, remote unlock and remote shutdown) and an automobile with on-board drive-by-wire functionality (e.g., key fob identification, normal operations and operation under hazardous road conditions).

**4.2.1. Normal Operations.** Normal operations of the Toyota Prius are similar to those of most modern automobiles. When the driver approaches the vehicle, the vehicle senses the key fob. In the case of the Prius, the driver does not have to depress a button on the fob to activate the automobile. If the Prius determines that the driver has the correct key fob, it goes into the pre-operational mode and unlocks the driver side door when the handle is pulled. The driver can then operate the vehicle.

**4.2.2. Hazardous Road Conditions.** Since the early 1970s, automobiles have been equipped with Anti-Lock Braking System (ABS). More recently some automobiles, such as the Toyota Prius, have been equipped with traction control systems to automatically correct for loss of traction. These systems can be thought of as a super-set of ABSs and are very effective. As in the case of an ABS, the first time that a driver experiences a traction control system, there is a feeling of complete loss of control of the automobile. When the traction control system is operating, the system overrides any action that the driver takes to protect the automobile and passengers. In this dissertation, all discussions about traction control are assumed to apply only to the Toyota Prius.

**4.2.3. Remote Shutdown.** Many automobile models have the ability to be remotely disabled. This is intended to assist in recovering a stolen automobile, but it has the

potential for misuse [41]. Of particular concern are models, such as the Toyota Prius, that automatically shift into park when the system is turned off.

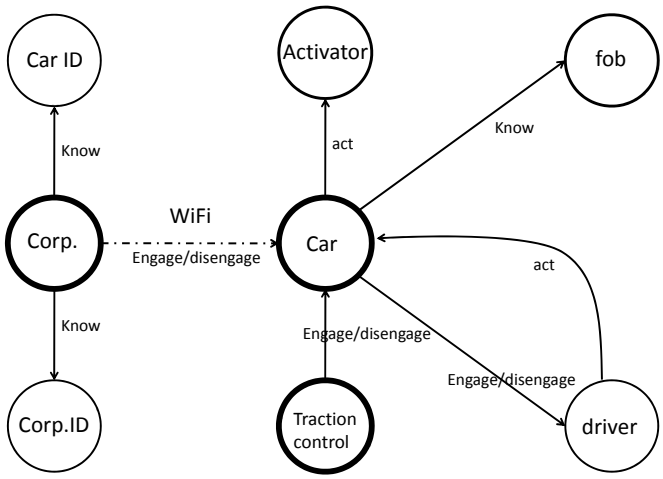



Figure 4.1. Schematic Operation of Corporation/Driver/Car Interactions

**4.2.4. Remote Operations.** If an automobile is equipped with OnStar, Toyota Safety Connect or some other similar service, the owner must trust the corporation or service provider. The specific limits of the service often cannot be discerned from the sales brochure. Also, it can be very difficult to determine if the actions undertaken by an automobile during its operation are due to traction control, remote corporate operations or a malfunction. In all three cases, the results can be the same the driver has no control over the automobile.

## 4.3. LEAKAGE OF RIGHTS

This section analyzes the leakage of rights using the security models described in the previous section.

### 4.3.1. Subjects, Objects, and Rights.

Subjects:

- $Car$ - The computerized control function of the vehicle.
- $Corporation$ - An entity, or its members, that may be able to control the vehicle through a wireless link; for example, Toyota Safety Connect[42]. Another example would be GM OnStar.
- $TractionControl$ ($TC$) - The vehicular system that takes control of the automobile under hazardous driving conditions. It includes the more familiar anti-lock braking system (ABS).

Objects:

- $Activator$: The operational network of the automobile, including physical operations such as brakes, shift and power on/off.
- $Fob$: The key fob that activates the automobile.
- $Driver$: The person sitting in the driver's seat of the automobile.
- $Car$ID: A unique identifier that allows remote commands to the automobile, including commands that change rights [43].
- $CorporateID$: A unique identifier that allows the corporation to securely sign a command sent to an automobile by $CarID$.
- These objects are attached to the $Activator$ and are operated by the $act$ right. These objects have no direct security impact. They exist in the model simply to make it easier to understand the object $Activator$.
  - Shift - The automobile will remain in PARK unless there is a valid key fob inside the vehicle. There is no physical shift control linkage.

- Brakes - The brakes will not operate under certain conditions, such as when the traction control is engaged. The vehicle decides at all times whether to brake or regenerate when the brake pedal is pressed.
- Accelerator - The accelerator is drive-by-wire and will not operate under certain conditions.
- Power On/Off - The ability to change the power state of the automobile.

Rights:

- $act$ (activate): The ability to perform an action, very much like a read, write, and execute.
- $know$: - Knowledge or possession of the $Fob$, $CorporateID$, or $CarID$. In the special case of the fob, the automobile acquires $know$ when the fob is in the proximity of the automobile and is remotely sensed. In the case of $Corporation$, $know$ can mean the information is retrieved from the files or received over a link of unspecified length and type.
- $engage/disengage$: The ability to engage or disengage control from another source. The automobile recognizes commands according to this hierarchy: $Corporation$, $TractionControl$, and $Driver$.

**4.3.2. Bell-LaPadula.** There are three security levels, CORPORATE, CAR and DRIVER. The most secure is the CORPORATE level and the least secure is the DRIVER level. The CAR environment is an intermediary between information held in the CORPORATE environment and the DRIVER (or unsecured) environment.

1. Corporation (highest level) which we refer to as [**CORPORATE**]
   - Subject $Corporation$
   - Object $CorporateID$

- Object $CarID$ (as $known$ by $Corporation$)

2. $Car$ system (Middle level) which we will refer to as [**CAR**]

- Subject $Car$
- Subject $TractionControl$
- Object $CarID$ (as $known$ by $Car$)
- Object $Activator$

3. $Driver$ (lowest level) which we refer to as [**DRIVER**]

- Object $Fob$
- Object $Driver$
- Object $CarID$ (as $known$ by $Fob$)

The security domains for the model create compartments that partition the system into domains that restrict actions between levels. This is particularly important when a high security subject lowers its security level to write to a lower security object.

1. [RASSIST] The security domain of the $Corporation$ and every $Car$ connected to the WiFi network.

- Subject $Corporation$
- Subject $Car$
- Object $CorporateID$
- Object $CarID$ (as $known$ by $Corporation$)

2. [CAN] The security domain of the actual Controller Area Network (CAN) of the target Prius.

- Subject $Car$
- Subject $TractionControl$

- Object $CarID$ (as $known$ by $Car$)
- Object $Activator$

3. [CABIN]

- Subject $Car$
- Object $Fob$
- Object $Driver$
- Object $CarID$ (as $known$ by $Fob$)

The Subjects and Objects within the security model are placed in the level and domain [LEVEL, (DOMAIN)], as shown in Table 4.1. Figure 4.1 presents the subject-object interactions: $Car$ receives the authorization to operate from $Fob$, $TractionControl$ or $Corporation$.

**Theorem 2.** *The BLP model allows $Car$ to operate under hazardous road conditions.*

*Proof.*

1. $Car$ senses hazardous road conditions. No security actions have occurred.
2. $Car$ uses the *****property to engage $TractionControl$.
   [CAR] ≤ [CAR] and (CAN,CABIN) ⊆ (CAN,CABIN)
3. $Car$ lowers its security level to [DRIVER] and deletes $act$ from $Driver$.
   [DRIVER] ≤ [DRIVER] and (CABIN) ⊆ (CAN,CABIN)
4. $Car$ restores its security level to [CAR, (CAN,CABIN)].
5. $TractionControl$ uses $act$ to control the automobile via $Car$ (and $Activator$).
   [CAR] ≤ [CAR] and (CAN) ⊆ (CAN,CABIN)
6. $Car$ senses normal road conditions. No security actions have occurred.
7. $Car$ uses the *****property to disengage $TractionControl$.
   [CAR] ≤ [CAR] and (CAN) ⊆ (CAN,CABIN)

8. $Car$ lowers its security level to [DRIVER] and restores $act$ over $Car$ to $Driver$.

   [DRIVER] ≤ [DRIVER] and (CABIN) ⊆ (CAN,CABIN)

9. $Car$ restores its security level to [CAR, (CAN,CABIN)].

10. $Driver$ uses *****property and $act$ to operate Prius.

    [DRIVER] ≤ [CAR] and (CABIN) ⊆ (CAN,CABIN) □

**Theorem 3.** *The BLP model allows for the $Car$ to operate under the remote control of Toyota.*

*Proof.*

1. $Corporation$ lowers its security level to [CAR].

2. $Corporation$ uses the *****property to send $engage/disengage$ command to $Car$.

   [CAR] ≤ [CAR] and (RASSIST) ⊆ (CAN,RASSIST, CABIN)

3. $Car$ lowers its security level to [DRIVER] and deletes $act$ from $Driver$.

   [DRIVER] ≤ [DRIVER] and (CABIN) ⊆ (CAN,CABIN)

4. $Car$ restores its security level to [CAR, (CAN,CABIN)].

5. $Car$ uses *****property to send $disable$ command to $TractionControl$.

   [CAR] ≤ [CAR] and (CAN) ⊆ (CAN)

6. $Corporation$ uses the *****property to control the $Car$ via $act$.

   [CAR] ≤ [CAR] and (RASSIST) ⊆ (CAN,RASSIST,CABIN)

7. $Corporation$ uses the *****property to return control to the $Car$.

   [CAR] ≤ [CAR] and (RASSIST) ⊆ (CAN,RASSIST,CABIN)

8. $Corporation$ restores its security level to [CORPORATE].

9. $Car$ lowers its security level to [DRIVER] and restores $act$ over $Car$ to $Driver$.

   [DRIVER] ≤ [DRIVER] and (CABIN) ⊆ (CAN,CABIN)

10. $Car$ restores its security level to [CAR, (CAN,CABIN)].

Table 4.1. BLP Security

| Subject | Level | | | Domain | | |
|---|---|---|---|---|---|---|
| Object | CORPORATE | CAR | DRIVER | RASSIST | CAN | CABIN |
| Activator | | x | | | x | |
| Fob | | | x | | | x |
| Driver | | | x | | | x |
| Car | | x | | x | x | x |
| CarID | x | x | x | x | x | x |
| Corporation | x | | | x | | |
| CorporateID | x | | | x | | |
| Traction Control | | x | | | x | |

11. $Driver$ can now use the *****property to write $act$ to the $Car$.

    $[\text{CABIN}] \leq [\text{CAR}]$ and $(\text{CABIN}) \subseteq$

    (CAN,RASSIST,CABIN) □

This allows the corporation to take control of any function of $Car$ at any time. While there exists a slight possibility that this might be useful to the driver in the situation of a runaway or stolen automobile, the security implications are more sinister in terms of an attack[41]. If an employee of the corporation were to use these commands maliciously, the implications may well be life threatening. A command could be entered remotely to set the accelerator to maximum and the driver and passenger would be helpless. The assumption is that this could happen but never would happen–it is simply a matter of trust.

**4.3.3. Lipner.** In order to implement the Lipner (Biba/Bell-LaPadula) model, we will use the same security levels and domains as in our existing BLP model of the $Car$. However, because we also consider the Biba model, it is necessary to define integrity levels and integrity compartments (**Table 4.2**). The integrity levels are:

- $[TRUSTED]$ - Completely trusted
  - Object$CorporateID$
  - Subject $Car$

  - Subject Traction Control
  - Object $Activator$
  - Object $CarID$
  - Object $Fob$
  - Object $Driver$

- $[UNTRUSTED]$ - Not trusted
  - Subject $Corporation$

The integrity compartments are:

- $[WiFi]$ - Wireless network
  - Subject $Corporation$
  - Subject $Car$
  - Object $CarID$
  - Object$CorporateID$

- $[CAN]$ - Controller Area Network
  - Subject $Car$
  - Subject $TractionControl$
  - Object $Activator$
  - Object $CarID$

- $[CABIN]$ - Passenger compartment
  - Object $Fob$
  - Object $Driver$

**Assumption 1.** *Ring Policy We will use the Ring Policy for our model.*

**Theorem 4.** *The Lipner model allows $Car$ to operate under normal conditions.*

*Proof.*

1. The Prius senses the fob. No security actions have taken place at this point.
2. The Prius reads the fob for the proper credentials. $Car$ is a subject and fob is an object. Any subject can read any object.
3. $Car$ writes to object Driver to grant $act$ over $Car$. $Car$ [TRUSTED,(CABIN)] is at the same integrity level as the object $driver$ [TRUSTED,(CABIN)].

□

**Theorem 5.** *The Lipner model allows $Car$ to operate under hazardous road conditions.*

*Proof.*

1. The $Car$ senses hazardous road conditions. This is allowed because no security actions are involved.
2. $Car$ engages Traction Control. Both are at the same integrity level [TRUSTED,(CAN)].
3. $Car$ deletes right $act$ over $Car$ from object $Driver$. A subject may write to an object only if the subject is at the same, or higher, integrity level as the object. $Car$ [TRUSTED,(WiFi,CAN,CABIN)] is the same integrity level as $driver$ [TRUSTED,(CABIN)].
4. Traction Control uses $act$ to control the automobile via $Car$ and $Activator$. A subject can only execute another subject if it is at the same or higher integrity level. Traction Control, $Car$, and $Activator$ are all at level [TRUSTED,(CAN)].
5. $Car$ senses normal road conditions. No security actions occur.
6. $Car$ disengages Traction Control. A subject can only execute another subject if it is at the same or higher integrity level. Traction Control and $Car$ are both at level [TRUSTED,(CAN)].

Table 4.2. Lipner Integrity Matrix

| Subject | Level | | Compartment | | |
|---|---|---|---|---|---|
| Object | TRUSTED | UNTRUSTED | WiFi | CAN | CABIN |
| Activator | x | | | x | |
| fob | x | | | | x |
| driver | x | | | | x |
| Car | x | | x | x | x |
| CarID | x | | x | x | x |
| Corporation | | x | x | | |
| CorporateID | x | | x | | |
| Traction Control | x | | | x | |

7. $Car$ gives right $act$ over $Car$ to object $Driver$. A subject may write to an object only if the subject is at the same, or higher, integrity level as the object. $Car$ [TRUSTED,(WiFi,CAN,CABIN)] is the same integrity level as $driver$ [TRUSTED,(CABIN)]. □

Before we look at remote operations, $Driver$ is the only person who can correctly assess the needs of the moment. A remote command, no matter how innocent or safe in theory, could be life threatening to the passengers. If a command is issued, $Driver$ must determine the source of the command ($Corporation$ or $TractionControl$), the current situation, and the proper actions to preserve the safety of $Car$. Therefore, from the perspective of the $Driver$, any actions that are not directly under the command of $Driver$ or $TractionControl$ are inherently not trusted. Indeed, they are most definitely threatening.

**Theorem 6.** *If $Corporation$ is not trusted, then the Lipner model does not allow $Car$ to operate under remote operations by $Corporation$.*

*Proof.*

1. $Car$ is operating normally under the control of $Driver$. This is allowed as no security actions are involved.
2. $Corporation$ sends $engage/disengage$ command to $Car$. A subject can only execute another subject if it is at the same or higher integrity level.

$Corporation$ [UNTRUSTED,(WiFi)] is at a lower integrity level than

$Car$ [TRUSTED,(WiFi,CAN,CABIN)] so the execution is not allowed (fails).

Therefore Lipner fails for this procedure. □

Because $Corporation$ is not trusted, the Lipner model stops $Corporation$ conducting remote operations. Unfortunately, this is not the way the automobile system functions in reality.

**4.3.4. Remarks about Applying the Lipner Model.** While the Prius works fairly well under BLP, it does not work as well under the Lipner model. Even though the Lipner Model deals better with issues of changing security levels, the Lipner model fails to allow the known functioning of the system when Toyota Safety Connect is operating.

**4.3.5. Noninference.** We will now examine how $Car$ behaves under Noninference with respect to $Driver$. We will use the same security levels as in the BLP Model (see Table 4.2).

**Definition 18.** *Terms*

*HIGH Level* $H$ = {*CORPORATE,CAR*}

*LOW Level* $L$ = {*DRIVER*}

We will use the commands and states listed in Table 4.3.

**Theorem 7.** *The Noninference model permits information flow to* $Driver$ *under hazardous road conditions.*

States: $\sigma_0, \sigma_{13}, \sigma_2, \sigma_9, \sigma_5, \sigma_7, \sigma_8, \sigma_{14}, \sigma_{12}, \sigma_{10}, \sigma_1, \sigma_0$

Commands($c_s$): $c_{13}, c_2, c_9, c_5, c_7, c_8, c_{14}, c_{12}, c_{10}, c_1$

To prove Noninference with respect to $Driver$, the following two properties must hold:

**Property.** $\tau$ is a valid trace:

*Proof.* $\tau = \{\sigma_0, \sigma_{13}, \sigma_2, \sigma_9, \sigma_5, \sigma_7, \sigma_8, \sigma_{14}, \sigma_{12}, \sigma_{10}, \sigma_1, \sigma_0\}$ □

Table 4.3. Commands and States

| Command | State |
|---|---|
| $c_0$: Null command | $\sigma_0$: $Car$ operating normally |
| $c_1$: $Car$ grants $Driver$ $act$ | $\sigma_1$: $Driver$ has $act$ rights |
| $c_2$: $Car$ grants $act$ to $TC$ | $\sigma_2$: $Car$ is in TC mode |
| $c_3$: $Corporation$ takes $act$ from $Car$ | $\sigma_3$: $Corporation$ controls the automobile |
| $c_4$: $Driver$ issues request to $Car$ | $\sigma_4$: $Car$ has a request from $Driver$ |
| $c_5$: $TC$ gives command to $Car$ | $\sigma_5$: $Car$ has a command from $TC$ |
| $c_6$: $Corporation$ issues command | $\sigma_6$: $Car$ has a $Corporation$ command |
| $c_7$: $Car$ issues command to $Activator$ | $\sigma_7$: $Activator$ has a command from $Car$ |
| $c_8$: $Activator$ executes command | $\sigma_8$: Car does something |
| $c_9$: $Car$ deletes $act$ from $Driver$ | $\sigma_9$: $Car$ is not under driver control |
| $c_{10}$: $TC$ relinquishes $act$ over $Car$ | $\sigma_{10}$: $TC$ is disabled |
| $c_{11}$: $Corporation$ releases control of $Car$ | $\sigma_{11}$: $Corporation$ releases $act$ over $Car$ |
| $c_{12}$: $Car$ senses normal conditions | $\sigma_{12}$: Road is normal |
| $c_{13}$: $Car$ senses dangerous conditions | $\sigma_{13}$: Road is dangerous |
| $c_{14}$: $Driver$ observes automobile's physical actions | $\sigma_{14}$: $Driver$ senses automobile's actions |
| $c_{15}$: $Car$ reads $Fob$ | $\sigma_{15}$: $Car$ has $CarID$ from $Fob$ |
| $c_{16}$: $Car$ notifies $Driver$ of success | $\sigma_{16}$: $Driver$ observes notification |
| | $\sigma_{17}$: $Car$ is parked and idle |

**Property.** $\tau \uparrow L$ is a valid trace

*Proof.* $\tau \uparrow L = \{\sigma_0, \sigma_8, \sigma_{14}, \sigma_0\}$

This is not a valid trace. If the high level command $c_2$ does not occur, then $Car$ will not move unless under the command of $Driver$ and state $\sigma_{14}$ will not occur. Therefore, the proof fails. Noninference does not hold with respect to $Driver$. □

In this case, the model is not Noninference secure with respect to $Driver$. Information that the automobile has engaged traction control is leaked to the driver.

**Theorem 8.** *The Noninference model permits information flow to $Driver$ under $Corporation$ remote operations.*

States: $\sigma_0, \sigma_3, \sigma_9, \sigma_{10}, \sigma_6, \sigma_7, \sigma_8, \sigma_{14}, \sigma_{11}, \sigma_1, \sigma_0$

Commands($c_s$): $c_3, c_9, c_{10}, c_6, c_7, c_8, c_{14}, c_{11}, c_1$

To prove Noninference with respect to $Driver$, we must prove two properties:

**Property.** $\tau$ is a valid trace:

*Proof.*

$\tau = \{\sigma_0, \sigma_3, \sigma_9, \sigma_{10}, \sigma_6, \sigma_7, \sigma_8, \sigma_{14}, \sigma_{11}, \sigma_1, \sigma_0\}$ □

**Property.** $\tau \uparrow L$ is a valid trace

*Proof.*

$\tau \uparrow L = \{\sigma_0, \sigma_8, \sigma_{14}, \sigma_0\}$ □

This is not a valid trace. If the high level command $c_3$ does not occur, the automobile remains under the control of the driver and state $\sigma_{14}$ can not be reached. **Proof fails - Noninference does not hold with respect to the** $Driver$**.**

**4.3.6. Remarks about Applying the Noninference Model.** The implications of information flow to $Driver$ under hazardous conditions and $Corporation$ remote control are not trivial. While $Driver$ knows that he or she is temporarily not in control of the Prius, it is not possible for $Driver$ to distinguish between the actions of the traction control mechanisms and remote commands. This is compounded by the fact that a system failure would give $Driver$ the same information. Thus, $Driver$ is left bewildered,confused and possibly frightened.

**4.3.7. Nondeducibility.** We now analyze how $Car$ behaves under Nondeducibility [44] with respect to $Driver$. Once again, we use the same security levels as in the BLP model.

**Definition 19.** *Terms HIGH Level:* $H$ = {*CORPORATE,CAR*} *and LOW Level:* $L$ = {*DRIVER*}.

Once again, we use the commands and states from Table 4.3.

**Theorem 9.** *The Nondeducibility model permits information flow to the driver under Normal Operations.*

The events required to begin operations of the automobile are:

States: $\sigma_{17}, \sigma_{15}, \sigma_1, \sigma_{16}, \sigma_4$

Commands($c_s$): $c_{15}, c_1, c_{16}, c_4$

To disprove Nondeducibility, we need to show that there is only one set of High Inputs and Low Inputs that could have produced the events as seen by the driver.

*Proof.* Proof by contradiction

Assume there are multiple sets of HI that when taken with LI produce the known trace.

1. These HI cannot come from $TractionControl$ because the $Car$ is not behaving mysteriously.
2. These HI cannot come from $Corporation$ for the same reason.
3. These HI must come from $Car$.
4. $Car$ has a limited number of possible HI it can produce:
    - delete $act$ over $Car$ from $Driver$ is not possible because the $Driver$ can operate the automobile
    - $engage$ Traction Control is not possible because the automobile is not behaving mysteriously
    - $Car$ grants $act$ over $Car$ to $Driver$ is possible because the $Driver$ can operate the automobile
    - $Car$ reads $Fob$ is possible because this is required to grant $act$ over $Car$ to $Driver$
5. No other actions are possible, therefore there is only one set of HI that leads to the known trace. Contradiction!

Therefore, the model is not Nondeducibility secure with respect to $Driver$ under Normal Operations. □

**Theorem 10.** *The Nondeducibility model does not leak information to the $Driver$ under hazardous road conditions (Traction Control Mode).*

States: $\sigma_0, \sigma_{13}, \sigma_2, \sigma_9, \sigma_5, \sigma_7, \sigma_8, \sigma_{14}, \sigma_{12}, \sigma_{10}, \sigma_1, \sigma_0$

Commands($c_s$): $c_{13}, c_2, c_9, c_5, c_7, c_8, c_{14}, c_{12}, c_{10}, c_1$

The trace would be: $T_r = \sigma_0, \sigma_{13}, \sigma_2, \sigma_9, \sigma_5, \sigma_7, \sigma_8, \sigma_{14}, \sigma_{12}, \sigma_{10}, \sigma_1, \sigma_0$ and $Tr_L = \sigma_0, \sigma_8, \sigma_{14}, \sigma_0$. In plain language, the driver is driving the automobile, suddenly the $Car$ begins to behave mysteriously, and then the $Car$ suddenly lets the driver operate the automobile again. We will prove hazardous road conditions is Nondeducibility secure in conjunction with $Corporation$ remote operations.

**Theorem 11.** *The Nondeducibility model does not leak information to the $Driver$ under $Corporation$ remote operations.*

States: $\sigma_0, \sigma_3, \sigma_9, \sigma_{10}, \sigma_6, \sigma_7, \sigma_8, \sigma_{14}, \sigma_{11}, \sigma_2, \sigma_1, \sigma_0$

Commands($c_s$): $c_3, c_9, c_{10}, c_6, c_7, c_8, c_{14}, c_{11}, c_2, c_1$

The trace would be: $Tr = \sigma_0, \sigma_3, \sigma_9, \sigma_{10}, \sigma_6, \sigma_7, \sigma_8, \sigma_{14}, \sigma_{11}, \sigma_2, \sigma_1, \sigma_0$ and $Tr_L = \sigma_0, \sigma_8, \sigma_{14}, \sigma_0$. In plain language, the driver is driving the automobile, it suddenly begins to behave mysteriously, and then $Car$ suddenly lets the driver operate the automobile again. We will prove $Corporation$ remote operations is Nondeducibility secure in conjunction with hazardous road conditions.

*Proof.* Hazardous road conditions and $Corporation$ remote operations are both Nondeducibility secure.

In both hazardous road conditions (Theorem 10) and $Corporation$ remote operations (Theorem 11), the driver experiences the exact same low level trace. The automobile is operating normally, the automobile behaves mysteriously and is not under the driver's

control, and suddenly things return to normal. By our definition, ND holds if:

$$ND(ES) = \forall \tau_L, \tau_H \in Tr$$

such that

$$\exists \tau \in Tr : \tau|L = \tau_L|L \wedge \tau|HI = \tau_H|H$$

Since we have shown two different traces, from hazardous road conditions and $Corporation$ remote operations, that have different High Inputs but the same $\tau_L$, it is not possible to deduce the High Inputs. Therefore, both are Nondeducibility safe for the driver. □

**4.3.8. Remarks about Applying the Nondeducibility Model.** From the viewpoint of the driver, it would be better if Nondeducibility ***did not hold*** for hazardous conditions and $Corporation$ remote operations. The driver has no control in either case and cannot determine the source of the strange actions of the $Car$. The driver may not be able to turn off the automobile and coast to a safe stop. Some newer automobiles shift into park when turned off.

## 4.4. REMARKS

Like most modern automobiles, the Toyota Prius as a drive-by-wire system can present some interesting security issues. The system fits the multi-level BLP model, but the requirements for a more secure subject to lower its security level to make the system function implies a large amount of trust in the entity. In the Lipner model the $Corporation$ subject is not trusted, resulting in the system operations being inconsistent with known operations. Both Noninference and Nondeducibility are information flow models that describe the ability of the driver to ascertain how the vehicle is being operated. Specifically, the system is Nondeducibility secure with respect to the driver which means the driver cannot ascertain if $Corporation$ remote operations or the $Car$ is controlling the behavior.

If the owner subscribes to OnStar, Toyota Safety Connect, or some similar service, the driver must trust the service. We have shown that in hazardous situations or in remote operations the driver is **not** in control of the automobile. There is nothing the driver can do in these situations but trust that all is well. Such concerns spread beyond the systems studied here; the recent Stuxnet worm [15] had a similar effect of blinding the system operator from the actual CPS operation. As such, this dissertation's analysis is indicative of the type of analysis needed before large-scale adoption of cyber-physical services.

This section models a particular CPS, the drive-by-wire automobile using a number of security models from BLP to the IFS models such as Noninterference, Noninference, and trace-based Nondeducibility. If a model is used that does not take into account IFS, the results do not clearly describe the observed actions of the CPS. The IFS models more closely reflect the reality of the drive-by-wire automobile, but there are still issues.

The models as presented are all trace based. The CPS must be closely followed through the actions of interest and *then* the trace can be examined to insure that the particular IFS holds. For both NI and ND the model must be run *twice* under different security conditions and the resulting traces compared. Trace based information flow models do not offer the promise of real-time analysis of CPS.

As noted in Section 2.3, Sutherland Nondeducibility can best be expressed as a modal frame based model rather than a trace based model. With frame based ND models, the requirement to run the model multiple times can be relaxed. This would seem to eliminate all of the major issues with ND but in many cases it does not. Sutherland's ND requires the model to contain potential valuation functions for *all Well-formed formula (wff) for all worlds.* This is often not the case with CPS. If valuation functions are missing, the model fails and ND cannot be determined one way or the other.

The next section will address these issues.

# 5. A MULTIPLE SECURITY DOMAIN MODEL OF A DRIVE-BY-WIRE SYSTEM

Traditional security models partition the security universe into two distinct and completely separate worlds: us and them. This partition is absolute and complete. More complex situations are most commonly treated as sets of increasingly more secure domains. This view is too simplistic for 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.We present a model that handles situations where the security domains are complex or the threat space is ill defined. To demonstrate our method, we examine a “drive by wire” system from both the traditional view and in light of the modern reality. This dissertation examines the system from the viewpoint of the driver with special emphasis on the driver’s inability to determine who, or what, is actually in control of the automobile during critical situations.

## 5.1. INTRODUCTION

It is natural to reduce the concept of security to “walling the bad guys out.” From primitive forts to sophisticated medieval castles to modern computer security systems, this model has held up reasonably well. Unfortunately, as situations become more complex, and the “bad guys” more astute, these models became less effective.

Models such as Bell-LaPadula(BLP) [45], Lipner [24], and Noninterference [25] work well for most situations as long as one is aware of their limitations. When viewed as increasingly more effective and sophisticated attempts to deal with the real world, these models serve for the everyday as long as there is something better for the more difficult and more demanding possible situations. These models depend upon clean and idealized axioms and, in general, require the ability to know the sequence of actions (trace) and are input total [46] [47]; i.e., we must know all the actions and their consequences to be able to

analyze the security situation. These models all forbid any dependencies between input [46] and frequently rely upon objects following unenforced rules such as BLP's "no write down". Each has their place, but for more demanding applications we need stronger tools.

IFS in CPS leads to particularly challenging and complex security domains. Most security models are composed of "secure" and "not secure". Unfortunately, this focus leaves these models open to attacks that do not steal information but simply disrupt critical information flow.

ND was introduced by Sutherland [8] in 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 disjoint sets and information is restricted to *one side of the partition or the other* [7]. 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 that simply cannot be evaluated.

In Section 5.4 we use both traditional Nondeducibility and Multiple Security Domain Nondeducibility to model a "drive by wire" automobile connected to a roadside assistance network such as General Motors OnStar or Toyota Connect.

> *Of prime concern is the simple question: can the driver determine when the car is under his/her control, the control of the on-board computer, or under the control of something outside the car? [1]*

Section 3.3 outlines the modal techniques and theory behind both security models. In Section 5.4 we model in detail: normal operations, hazardous road conditions (a.k.a traction control), and corporate remote control of the car.

## 5.2. PROBLEM STATEMENT

Is it reasonable to use computer security models to describe a CPS where the attack is designed to disrupt safe operations by concealing critical information flows? Specifically,

Table 5.1. Definition of State Variables

| Variable | |
|---|---|
| $s_0$ | Car is behaving normally($\top$) |
| $s_1$ | $driver$ is aware of car's behavior |
| $s_2$ | $car$ is accepting commands from $driver$ |
| $s_3$ | $car$ is accepting commands from $tc$ |
| $s_4$ | $car$ is accepting commands from $corp$ |
| $s_5$ | $car$ is faulty and not accepting commands |

can the drive-by-wire automobile connected to a road side assistance network be described correctly using Sutherland's Nondeducibility or MSDND?

## 5.3. SPECIFIC CASE OF THE DRIVE-BY-WIRE AUTOMOBILE

This section will use the same specific model of the drive-by-wire automobile introduced in Section 4. The same subjects, objects, and modes of will be examined using a Kripke frame based model.

Depending on which mode the car is in, the driver may not be able to distinguish who or what is actually in control. Of particular interest is remote operation by *corp* which exists in one security domain v.s. operation by *driver* in another security domain. What the driver can and cannot ascertain is governed by the information flow that exists among domains, both in the cyber, and in the physical.

The ensuing discussion shows how classic models of information flow and Deducibility break down in the cyber-physical environment. This dissertation develops a multiple security domain model and applies it to the car model.

## 5.4. SPECIFIC EXAMPLE OF THE DRIVE-BY-WIRE PRIUS

**5.4.1. Structure of the Model.** We will limit our discussion to the state variables given in Table 5.1 and 5.2.

Table 5.2. Logical Statements of Interest

| $\varphi_i$ | state | |
|---|---|---|
| $\varphi_0$ | $s_0$ | The car is behaving normally |
| $\varphi_1$ | $s_1$ | $driver$ is aware of car's behavior |
| $\varphi_2$ | $s_2$ | The $driver$ is in command |
| $\varphi_3$ | $s_3$ | Traction Control is in command |
| $\varphi_4$ | $s_4$ | The $corp$ is in command |
| $\varphi_5$ | $s_5$ | The $car$ is not working correctly |
| $s_d$ | $d = \top$ | $d = \varphi_2 \wedge \sim\varphi_3 \wedge \sim\varphi_4 \wedge \sim\varphi_5$ |
| $s_t$ | $t = \top$ | $t = \sim\varphi_2 \wedge \varphi_3 \wedge \sim\varphi_4 \wedge \sim\varphi_5$ |
| $s_c$ | $c = \top$ | $c = \sim\varphi_2 \wedge \sim\varphi_3 \wedge \varphi_4 \wedge \sim\varphi_5$ |
| $s_f$ | $f = \top$ | $f = \sim\varphi_2 \wedge \sim\varphi_3 \wedge \sim\varphi_4 \wedge \varphi_5$ |

Table 5.3. Valuation Functions of the Model

| Valuation | Result |
|---|---|
| $\mathbb{V}_0^i(w) = s_0 \wedge \top$ | "true" $\leftrightarrow$ $car$ is behaving normally |
| $\mathbb{V}_1^i(w) = s_1 \wedge \top$ | "true" $\leftrightarrow$ $driver$ knows he is in control |
| $\mathbb{V}_2^i(w) = s_2 \wedge \top$ | "true" $\leftrightarrow$ $driver$ is in control of $car$ |
| $\mathbb{V}_3^i(w) = s_3 \wedge \top$ | "true" $\leftrightarrow$ $tc$ is in control of $car$ |
| $\mathbb{V}_4^i(w) = s_3 \wedge \top$ | "true" $\leftrightarrow$ $corp$ is in control of $car$ |
| $\mathbb{V}_5^i(w) = s_3 \wedge \top$ | "true" $\leftrightarrow$ $car$ is in a failure state |

We will now define a set of logical conditions, $\varphi_i, d, t, c, f$, that we can evaluate to determine how the car is responding to commands, see Table 5.2.

Similarly, we can define valuation functions for some of the state variables in the frame as given in Table 5.3. On any given world, these valuation functions will return the value of the corresponding state variable as seen by the entity in control $i \in \{d, t, c, f\}$. Either the driver $d$, traction control $t$, or corporation $c$ is in control or the car is faulty $f$ and nothing is in control.

From observing the actual operation of the car, there is an obvious constraint.

$$\mathbb{V}_i^t(w) = s_i$$
$$\mathbb{V}_i^c(w) = s_i$$

$$\mathbb{V}_i^d(w) = \begin{cases} s_i & i < 3 \\ (s_3 \vee s_4 \vee s_5) & \text{otherwise} \end{cases}$$

Figure 5.1. Evaluation Functions for the Drive-by-Wire Car

Table 5.4. Possible Worlds $w_i$ for the Drive-by-Wire Car

| world | in control | $s_2$ | $s_3$ | $s_4$ | $s_5$ |
|---|---|---|---|---|---|
| $w_2$ | $d$ | $\top$ | $\bot$ | $\bot$ | $\bot$ |
| $w_3$ | $t$ | $\bot$ | $\top$ | $\bot$ | $\bot$ |
| $w_4$ | $c$ | $\bot$ | $\bot$ | $\top$ | $\bot$ |
| $w_5$ | $f$ | $\bot$ | $\bot$ | $\bot$ | $\top$ |

**Constraint** (The $car$ can allow only one source of commands, $control_i$ at a time)**.** For some arbitrary world, $w \in W$, this can be expressed by the following set of conditions:

$$w \vDash d \quad \leftrightarrow \quad w \vdash \Box\sim(t \vee c \vee f)$$
$$w \vDash t \quad \leftrightarrow \quad w \vdash \Box\sim(c \vee f \vee d)$$
$$w \vDash c \quad \leftrightarrow \quad w \vdash \Box\sim(f \vee d \vee t)$$
$$w \vDash f \quad \leftrightarrow \quad w \vdash \Box\sim(d \vee t \vee c).$$

This constraint can be expressed as the predicate which evaluates to $1$ if that entity is in control and $0$ otherwise:

$$w \vDash d \quad \leftrightarrow \quad control_d = control_1 = 1$$
$$w \vDash t \quad \leftrightarrow \quad control_t = control_2 = 1$$
$$w \vDash c \quad \leftrightarrow \quad control_c = control_3 = 1$$
$$w \vDash f \quad \leftrightarrow \quad control_f = control_4 = 1$$
$$w \vdash \Box\left(\sum_{i=1}^{4} control_i = 1\right).$$

**5.4.2. The Sutherland Nondeducibility Model.** It would be of no real interest to re-examine all of the same modes of operations of the drive-by-wire automobile using Sutherland ND again. However, there is a key difference between the trace based analysis of Section 4 and a modal based analysis. The examination of one mode of operation will suffice to illustrate this difference.

Consider how the car behaves under the Sutherland Model with respect to the $driver$ during hazardous road conditions, i.e. traction control. The worlds to be considered are given in Table 5.4. The evaluation functions for *right* domain elements $tc$ and $corp$ are identical but the evaluation function for the *left* element $driver$ must reflect the lack of access to *right* level entities. A valid set of evaluations is given in Figure 5.1.

**5.4.3. Hazardous Road Conditions.** When the road conditions deteriorate and traction control takes over, the driver may be startled.

**Theorem 12.** *The Sutherland model prevents information flow to the $driver$ under hazardous road conditions.*

*Proof.* When the car senses hazardous road conditions, control is automatically transferred from $driver$ to $tc$. The driver, and passengers, can still sense the actions of the car due to the cyber-physical nature of the entire system but cannot evaluate what is causing the car to do what the driver senses. Using the worlds, states, and evaluation functions we have previously defined (see Tables 5.1 and 5.4 and Figure 5.1) we see:

$$\mathbb{V}^d_2(w_3) = \mathbb{V}^d_2(w_5) = (s_2 = \bot)$$
$$\mathbb{V}^d_3(w_3) = \mathbb{V}^d_3(w_5) = \top$$
$$\mathbb{V}^d_5(w_3) = \mathbb{V}^d_5(w_5) = \top$$
$$\mathbb{V}^t_3(w_3) \neq \mathbb{V}^t_3(w_5)$$
$$\mathbb{V}^t_5(w_3) \neq \mathbb{V}^t_5(w_5)$$

From the viewpoint of the $tc$(*right*):

$$\mathbb{V}^t_2(w_3) = \mathbb{V}^t_2(w_5) \wedge (\mathbb{V}^t_3(w_3) \neq \mathbb{V}^t_3(w_5))$$

From the viewpoint of the $driver$(*left*):

$$\mathbb{V}_2^d(w_3) = \mathbb{V}_2^d(w_4) = \mathbb{V}_2^d(w_5)$$
$$\wedge(\nexists\, \mathbb{V}_3^d(w)) \wedge (\nexists\, \mathbb{V}_4^d(w))$$
$$\wedge(\nexists\, \mathbb{V}_5^d(w))$$

When $tc$ *is in control* is secure from the driver, because the driver lacks valuations $\mathbb{V}_3^d(w), \mathbb{V}_4^d(w)$, and $\mathbb{V}_5^d(w)$. But Sutherland's Nondeducibility *cannot even be evaluated* because there are *no* valuation functions for the driver to determine exactly what events are hidden. The traction control module has access to all the valuations to determine ND, but the driver does not. Sutherland's Nondeducibility cannot properly describe the CPS in this situation. □

**5.4.4. Remarks about Applying the Sutherland Nondeducibility Model.** Before, in Section 4.3.7, Sutherland Nondeducibility was demonstrated by comparing the resulting traces as seen by $driver$ in two different scenarios Because the traces were identical, Sutherland Nondeducibility holds. This procedure could not yield ND(ES) without examining both traces. However, when Sutherland Nondeducibility is used as originally introduced in the modal version, Sutherland Nondeducibility can be shown directly from one scenario *if the subject has access to the valuation functions required.* When this is not the case, as in our current model, Sutherland Nondeducibility breaks down. This critical issue is what led to the development of Multiple Security Domains Nondeducibility.

**5.4.5. Multiple Security Domains Nondeducibility Model.** We will now examine how the model behaves when we take overlapping security domains into account, see Figure 5.2.

**5.4.6. Normal Operations.** First, how does the model resemble the actual CPS under normal conditions?

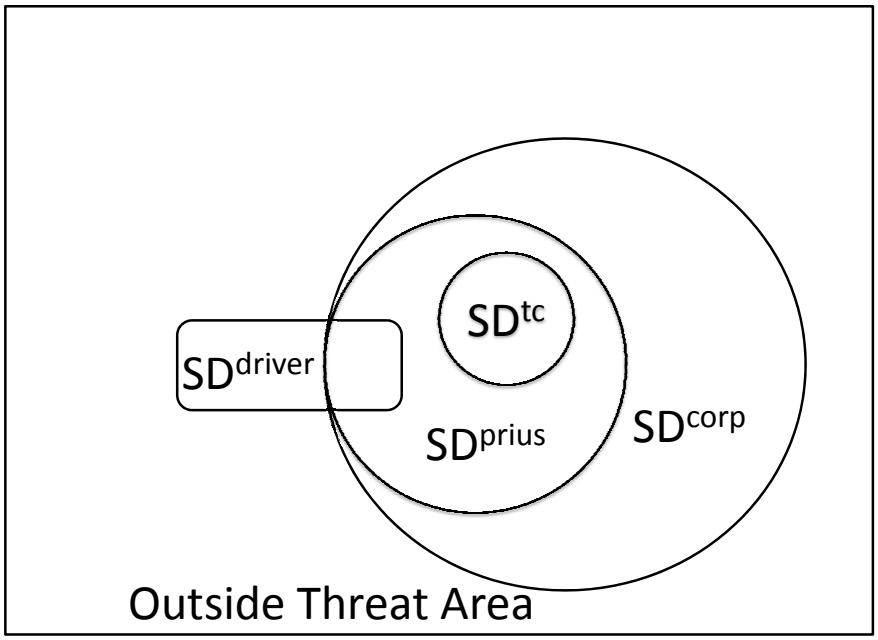



Figure 5.2. Security Domains in the Model

**Theorem 13.** *The MSDND model permits information flow to the driver under normal operations.*

Informally, $car$ responds to the $driver$ actions and this ensures that the $driver$ controls $car$. Under Normal Operations, MSDND(ES) does not hold as information has leaked from the security domain of the car to the security domain of the driver. Again, this is the desired result.

**5.4.7. Hazardous Road Conditions.** Next, how does the model resemble the actual CPS when road conditions deteriorate?

**Theorem 14.** *TheMSDND model yields Nondeducibility, thereby stopping critical information flow to the driver under hazardous road conditions.*

Under hazardous conditions, the $car$ acts exactly as in the Sutherland Model theorem 12. Using the worlds, states, and evaluation functions we have previously defined

(see Tables 5.1 and 5.4 and Figure 5.1) we see:

*Proof.* Given: The $driver$ knows something else is controlling the car and constraint still holds.

1. $\exists w \in W : w \vDash \sim d$ — $driver$ is not in control here
2. $w \vdash \Box\left(\sum_{i=1}^{4} control_i = 1\right)$ — something must be in control
3. $w \vdash \Box\left(\sum_{i=2}^{4} control_i = 1\right)$ — $tc$, $corp.$, or broken
4. $\mathbb{V}_2^d(w) = (s_2 = \perp)$ — $driver$ sees car's actions
5. $w \vDash\!\!\!\!/\ \mathbb{V}_3^d(w)$ — $driver$ can't tell it's $tc$
6. $w \vDash\!\!\!\!/\ \mathbb{V}_4^d(w)$ — is it $corp.$?
7. $w \vDash\!\!\!\!/\ \mathbb{V}_5^d(w)$ — is it broken?
8. Combining statements 3, 5, and 7 we obtain:

$$\text{MSDND(ES)} = \exists w \in W : \left[w \vdash \Box\left(\sum_{i=2}^{4} control_i = 1\right)\right] \wedge \left[w \vDash (\not\vDash \mathbb{V}_3^d(w) \wedge \not\vDash \mathbb{V}_5^d(w))\right]. \quad (5.1)$$

□

The $driver$ has a problem. In the domain $SD^d$ the physical actions of the car can be deduced, but the only deduction $driver$ can make is that he or she is not in control of the car. Strictly speaking, $driver$ does not have all the needed valuation functions and cannot even evaluate Sutherland ND(ES). Using the MSDND(ES) definition, the driver can correctly determine Nondeducibility. The driver can correctly determine he is not in control, but cannot determine exactly what is in control.

**5.4.8. Corporate Remote Operations.** When corporate, or some other entity, takes control remotely, the CPS behaves much as it does under poor road conditionis.

**Theorem 15.** *TheMSDND model yields Nondeducibility, thereby stopping critical information flow to the driver during remote operations.*

Under corporate remote operations, $car$ behaves as before, see theorem 11. Using the worlds, states, and evaluation functions we have previously defined (see Tables 5.1 and 5.4 and Figure 5.1) we see:

*Proof.* Given: The $driver$ knows something else is controlling the car and constraint still holds.

1. $\exists w \in W : w \vdash \sim d$ — $driver$ is not in control here
2. $w \vdash \Box\left(\sum_{i=1}^{4} control_i = 1\right)$ — something must be in control
3. $w \vdash \Box\left(\sum_{i=2}^{4} control_i = 1\right)$ — $tc$, $corp.$, or broken
4. $\mathbb{V}_2^d(w) = (s_2 = \perp)$ — $driver$ sees car's actions
5. $w \models\!\!\!\not\; \mathbb{V}_3^d(w)$ — $driver$ can't tell it's $tc$
6. $w \models\!\!\!\not\; \mathbb{V}_4^d(w)$ — is it $corp.$?
7. $w \models\!\!\!\not\; \mathbb{V}_5^d(w)$ — is it broken?
8. Combining statements 3, 6, and 7 we obtain:

$$\text{MSDND(ES)} = \exists w \in W : \left[w \vdash \Box\left(\sum_{i=2}^{4} control_i = 1\right)\right] \wedge \left[w \models (\not\models \mathbb{V}_4^d(w) \wedge \not\models \mathbb{V}_5^d(w))\right]. \quad (5.2)$$

□

**5.4.9. Remarks about Applying the Multiple Security Domains Model.** From the physical actions of the car, it is correct to deduce that the driver is not in control. What is in control is MSDND(ES) secure from the driver. Hazardous Conditions (traction control), Remote Corporate Operations, and possible mechanical failure all present the same way

to the driver and passengers. The longer this situation continues the more likely it is that something bad will happen.

## 5.5. REMARKS

The traditional view of security, the idea of "walling the bad guys out", is too simplistic. Viewing security domains as wholly contained within a threat space or within a less secure domain is inadequate as are the tools available. Restricting models to idealized partitions does not work well with cyber physical systems.

We have shown multiple security domains, without the necessity of ideal partitions, is a more realistic model. We have shown that in CPS information leaks throughout the model by observation of the physical actions of the system. Our new definition of MSDND(ES) can model traditional Nondeducibility as well as provide a definition of Nondeducibility that holds in CPS. Specifically, MSDND(ES) can easily model situations where critical information flow from one security domain to another is disrupted or denied altogether as in the Stuxnet worm attack.

We applied our model to a specific CPS, a drive-by-wire automobile, under real world conditions. Our model fits the CPS better than traditional Nondeducibility because it does not require us to partition the system into idealized domains that do not allow information flow between domains. Indeed, our model does not even need to address how the security domains interact once they have been properly defined. We have shown that we can relax the requirements of absolute domain partitioning and still model the system.

Furthermore, we have shown that since MSDND(ES) does not depend upon the ability to evaluate information flow between distinct and absolute partitions, our model does not require building complicated decision variables nor does it require access to the total input/output of the model. By relaxing the boundary conditions of the model, results are obtained by modal methods.

# 6. STUXNET TYPE ATTACKS ON CPS AND TRUST

Multiple Security Domains Nondeducibility, MSDND(ES), yields results even when the attack hides important information from electronic monitors and human operators. Because MSDND(ES) is based upon modal frames, it is able to analyze the event system as it progresses rather than relying on traces of the system. Not only does it provide results as the system evolves, MSDND(ES) can point out attacks designed to be missed in other security models.

This work examines information flow disruption attacks such as Stuxnet and formally explains the role that implicit trust in the cyber security of a cyber physical system (CPS) plays in the success of the attack. Modal operators are defined to allow the manipulation of belief and trust states within the model. We show how the attack hides and uses the operator's trust to remain undetected. In fact, trust in the CPS is key to the success of the attack.

## 6.1. INTRODUCTION

The advent of APT attacks [14] such as Stuxnet [48] have made older, traditional security models obsolete. The idea of reducing information security to walling the bad guys out is still valuable for a first line of defense, but as situations become more complex and the attacks become more sophisticated, these models become less effective. To make matters worse, these models depend upon clean, idealized axioms and require knowledge of the sequence of actions (trace). Therefore they are input total [46] [47]; i.e., we must know all the actions and their consequences to be able to analyze the security situation. These models are designed to prevent the theft and transfer of information to the outside world and are of limited use when the attack seeks only to hide critical information and not steal it. Stuxnet-like attacks require different tools.

Information flow security in CPS can lead to particularly complex security partitions. 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. A persistent attacker with enough time and backing will get in.

Models based upon modal logic and Kripke frames show promise in our efforts to understand these attacks. Modal logic techniques provide new ways to think about trust, information flow, and security domains. As modal logic is concerned with ways to view the truth of situations, we can look at how trust affects the models. Using Kripke frames we can get beyond traces and the need to look at the total input and output of the system. We need no longer wait until we can analyze the entire evolution of the model. With these models we can ask about the truth of what is presented and whether or not the results are valid, not just "is security preserved?" We will use these methods to examine APTs [14], or Stuxnet-like attacks.

Nondeducibility(ND) was introduced by Sutherland [8] as an attempt to use modal techniques and frames to model secure information in a partitioned model. The possible worlds of this model are partitioned into two or more disjoint sets in a step-wise manner. These sets are usually labeled as *high* and *low* with all information restricted to *one side of the partition or the other* [7]. Information that could not be gleaned from the other side of the partition was determined to be Nondeducibility secure. With this model, many sophisticated real world security issues could be effectively modeled and studied. However, the partition must be absolute and it must be simplistic. Overlapping security domains present severe difficulties for Sutherland's Nondeducibility as do information flows we simply cannot evaluate. MSDND can model Sutherland's Nondeducibility over any ES so this dissertation will concentrate on MSDND(ES).

We will use BIT logic [9] to show why a Stuxnet type attack is so difficult to discover in CPS. The doxastic logic of belief, information transfer, and trust is integral to the ability of a Stuxnet type attack to succeed and explains to an extent one of the basic reasons CPS are so vulnerable to these attacks [10].

## 6.2. PROBLEM STATEMENT

Can the role of implied trust in Stuxnet style attacks on CPS be formally modeled? Can we detect such an attack while it is in progress? How do we protect ourselves from something we trust? Most security efforts to date have been to wall the bad guys out and keep them from "seeing" or "stealing" sensitive information, but what if the goal is to hide critical information from the operator, i.e. the centrifuge is running at the wrong speed? If incorrect but reasonable information is sent, how will one know? Can one know? Trust in CPS monitors can be used to blind the human operator to the reality of APT attacks. This paper presents a generic method to guard against using trust to hide malicious actions.

## 6.3. THE ORGANIZATION OF THIS SECTION

Section 3.4 presents a brief explanation of BIT logic, Kripke frames, and models over Kripke frames. BIT logic is a tool for reasoning about the trust and belief key in Stuxnet type attacks.

An understanding of the underlying Kripke structure is key to understanding Nondeducibility from a modal viewpoint. Traditionally, Nondeducibility of information flows are examined from a trace base viewpoint. We will contrast the benefits of traces verses models. We will then give a description of our modal logic and the logic of belief, information transfer, and trust in these attacks.

In Section 6.4, we will describe our model of a specific CPS of a centrifuge/PLC monitored by an electronic system. We will also show how such an attack occurs and that such an attack is MSDND(ES) [2] secure. We will also show that without physical monitors to verify cyber-physical monitors, CPS are vulnerable to novel unexpected attacks. We will

show how belief in the readings from the CPS is critical to the success of an attack. We will further show the roles trust and belief play in attacking a CPS. Lastly, we will present some concluding remarks.

## 6.4. STUXNET-LIKE ATTACK MODELS

MSDND(ES) is particularly well suited to model attacks where the goal is to hide critical information from an operator rather than to steal or modify the information. There are two basic ways to hide this information: make it impossible to evaluate the desired question $\varphi$, or to disrupt the actual valuation function to return an unreliable valuation of the question $\varphi$. Trace based Nondeducibility is unable to properly handle this kind of attack and traditional Nondeducibility does not address the situation where the question cannot be evaluated at all.

## 6.5. CENTRIFUGE SYSTEM FUNCTIONAL MODEL

Consider a centrifuge used to enrich uranium. Uranium gas is passed into the centrifuge which must spin at a narrow range of frequencies in order to produce enriched uranium [49] [50] [51]. Such a device is usually controlled by a PLC device which is monitored by a PC running special software. Periodically the PC queries the controller as to how fast the centrifuge is spinning and adjusts the speed if it is outside the operational range. If the centrifuge speed is too far outside the range, the device could literally spin itself apart.

Consider the centrifuge system as divided up into multiple security domains defined in Table 6.1, see Figure 6.2. For simplicity, consider any buffers and communications channels to be in the PLC security domain $SD^1$. Let $\varphi$ be "The centrifuge is spinning within the desired range." Obviously either $\varphi$ or ~$\varphi$ must be true at all times. Under normal conditions, the PLC - PC system will monitor the centrifuge and make adjustments to insure that $\varphi$ is *true*. Under normal conditions, $\varphi$ is not MSDND(ES) secure.

Table 6.1. Centrifuge System Security Domains

| Domain | Valuation | |
|---|---|---|
| $SD^0$ | $\mathbb{V}^0_\varphi$ | Physical centrifuge |
| $SD^1$ | $\mathbb{V}^1_\varphi$ | Stuxnet-like virus |
| $SD^2$ | $\mathbb{V}^2_\varphi$ | PLC |
| $SD^3$ | $\mathbb{V}^3_\varphi$ | Monitor Station |
| $SD^4$ | $\mathbb{V}^4_\varphi$ | Human Operator |
| $SD^5$ | $\mathbb{V}^5_\varphi$ | Outside Observer |

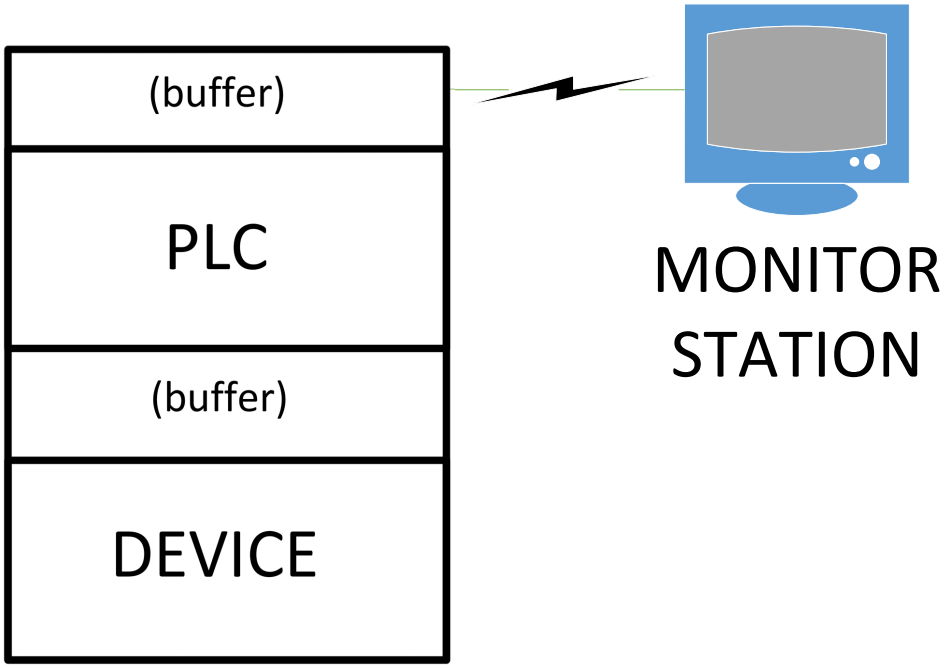



Figure 6.1. Centrifuge and PLC

## 6.6. CENTRIFUGE SYSTEM ATTACK MODEL

If a virus could be introduced into the PLC itself, the virus could easily disrupt the operation of enriching uranium [49] [50] [51]. The virus would be especially hard to detect

because it would not attempt to report the centrifuge speed to a location on the INTERNET; it would simply insure that the centrifuge was operating in a range that would not produce enriched uranium. In reality, the Stuxnet attack we are interested in was very simple. After infecting the PLC, the virus entered a passive phase where it recorded the messages between the PLC and monitor by intercepting the messages in the PLC communication buffer [52]. After a short period of time, the virus would acknowledge control messages from the monitor station and allow the centrifuge to spin at random frequencies outside its operational range. To a human operator, any queries via the monitor station would return positive results but the uranium produced would not be enriched enough to be useful. If the centrifuge lacked a physical speed indicator or the human operator did not happen to monitor the physical read out when the centrifuge was outside the optimal frequency range, the attack could go on until quality control tested the uranium or the centrifuge failed. Because the PLC would have reported valid frequency readings, it would be difficult to determine the cause of the failure.

The centrifuge is monitored and controlled directly by the PLC which is securely linked to a PC Monitoring station. The system is overseen by a human operator and superiors. Because this is a CPS, the actions cannot easily be hidden from outside observation at $SD^5$.

We will assume an attack by an APT much like Stuxnet. How the virus is introduced is not going to be discussed in this dissertation, but there are any number of ways the system could be successfully infected by a stray USB device, an infected piece of software, or contact with an network attached device. For example, a PC connected to the INTERNET might be infected from a website. The virus might then migrate via printer or media sharing to the monitor station on the secured network. The virus could easily migrate from there to the PLC.

**6.6.1. A Detailed Examination of the Attack Model.** Let: $\varphi$ be *true* if the centrifuge is operating within the desired frequency range and *false* otherwise. We define the security domains as in Table 6.1. During the recording phase of the attack (see Figures 6.1 and

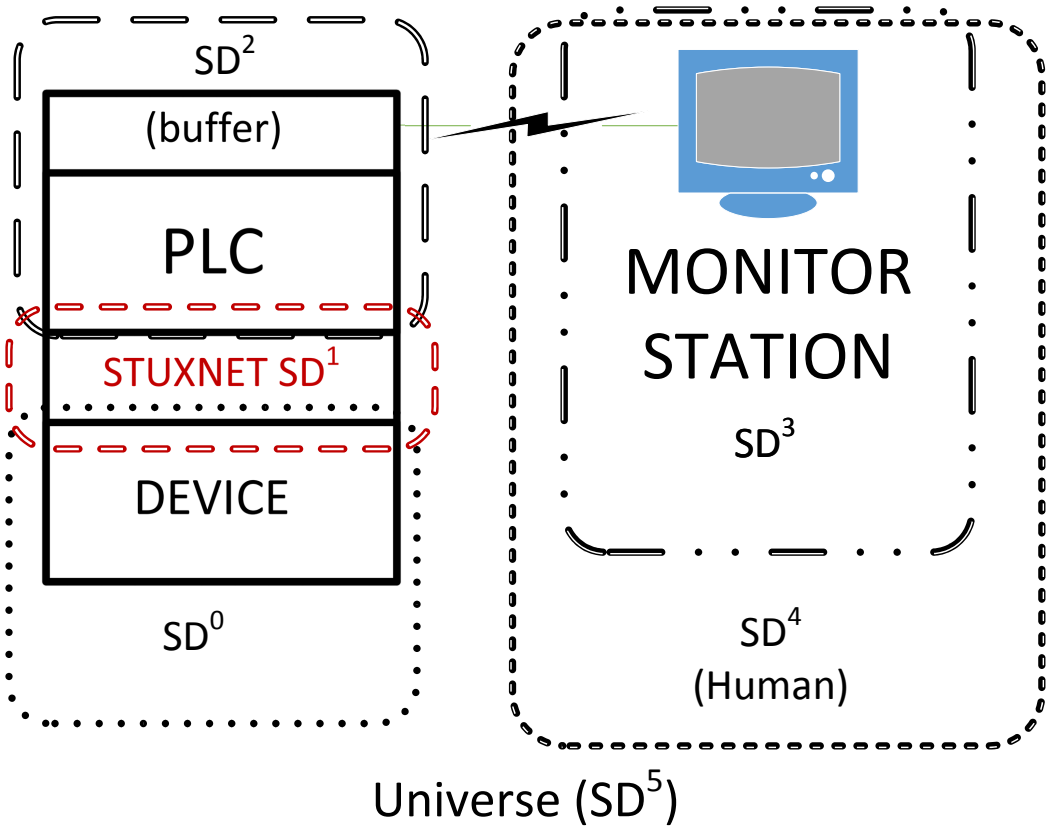



Figure 6.2. Security Domains

6.3) the system operates normally. When the system is under attack, see Figure 6.4, $\varphi$ is MSDND secure in some domains but not in others. This is the heart of the attack.

**Theorem 16.** *The speed of the centrifuge is not MSDND(ES) secure under normal operations and during recording phase.*

*Proof.*

**Case 16.1.** *Uninfected*

If the system is not infected, $\mathbb{V}_{\varphi}^{i}$ will be correctly evaluated for all domains.

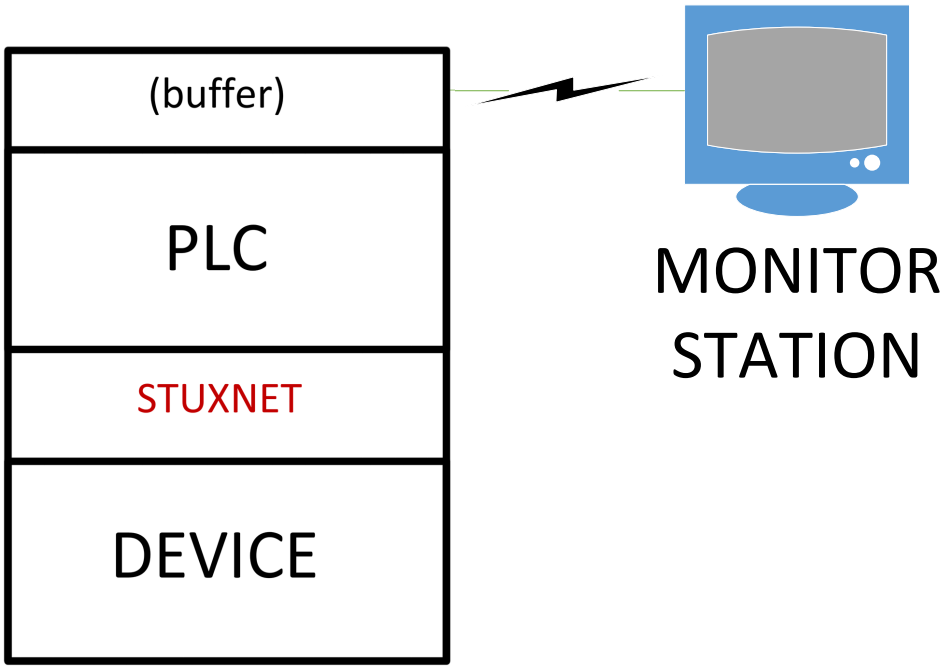



Figure 6.3. Centrifuge and PLC Controller Record Phase

**Case 16.2.** *Infected*

If the system is infected and Stuxnet is in the recording phase, all messages are recorded and then relayed. Therefore $\mathbb{V}^i_\varphi(w)$ will be correctly evaluated for all domains. Recording the message before it is relayed is actually problematic for Stuxnet as the delay leaks information about the attack. The effect is negligible, but if the centrifuge/PLC system is closely monitoring the time required to deliver a message from the sensor to the PLC itself the attack might be detected. □

**Theorem 17.** *The speed of the centrifuge is not MSDND(ES) secure at the physical centrifuge during the attack phase for infected systems.*

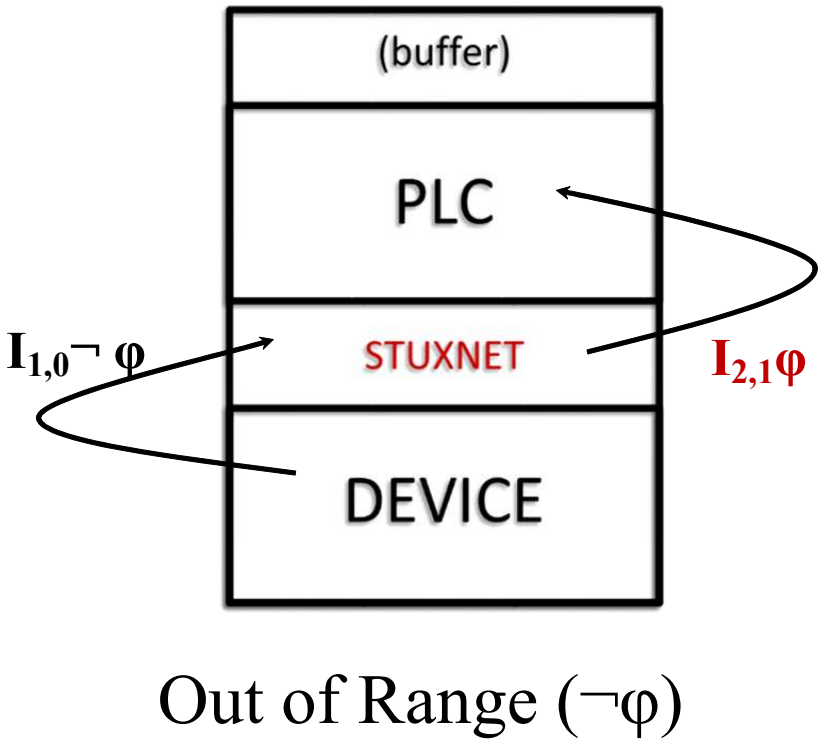



Figure 6.4. Centrifuge and PLC Controller Attack Phase

During the attack phase, the centrifuge correctly reports its status. The physical speed of the centrifuge is not MSDND(ES) secure from sensors on the centrifuge during the attack phase.

*Proof.*

To clarify notation, let $s_1$ be the state in which the centrifuge is within nominal bounds ($\varphi = \top$) and let $s_2$ be the state in which the centrifuge is not operating within nominal bounds ($\varphi = \bot$ and $\sim\varphi = \top$).

To show MSDND(ES), we must find a world $w$ such that:

$$MSDND(ES) = \quad \exists w \in W : w \vdash \Box\left[(s_1 \vee s_2) \wedge \sim(s_1 \wedge s_2)\right]$$
$$\wedge\left[w \vDash (\nexists\, \mathbb{V}^i_{s_1}(w) \wedge \nexists\, \mathbb{V}^i_{s_2}(w))\right]$$

It is obvious that $(s_1$ **xor** $s_2)$ = *true*, so the first condition for MSDND(ES) is met. However, the sensor directly measures the speed of the centrifuge and therefore both $\mathbb{V}^0_{s_1}(w)$ and $\mathbb{V}^0_{s_2}(w)$ are correctly evaluated for any $w$ and the conditions for MSDND(ES) are not met. □

**Corollary 1.** *The speed of the centrifuge is not MSDND(ES) secure in* $SD^i$ *if* $i$ *has direct access to* $\mathbb{V}^0_{\varphi}(w)$.

*Proof.*

| | | | |
|---|---|---|---|
| 1) | $w \vdash \Box[\varphi$ **xor** $\sim\varphi]$ | Definition of a boolean wff |
| 2) | $w \vDash \mathbb{V}^0_{\varphi}(w)$ | Theorem 6.6.1 |
| 3) | $\exists w \vDash \mathbb{V}^0_{\varphi}(w) \equiv \exists w \vDash \mathbb{V}^i_{\varphi}(w)$ | Entity $i$ has access to $SD^0$ |
| 4) | $w \vDash (\exists \mathbb{V}^i_{\varphi}(w) \wedge \nexists\, \mathbb{V}^i_{\sim\varphi}(w))$ | by step 3 |

Therefore, by definition MSDND does not hold. □

Corollary 6.6.1 implies that a physical reading may be used to break an MSDND attack that focuses on the chain of cyber monitoring reports. However, this is only useful if the physical monitoring cannot be compromised, e.g. a completely physical meter is available, and an entity does not believe and trust the cyber monitoring reports.

**Corollary 2.** *An entity* $i$ *will not believe a false report of the speed of the centrifuge if the entity has direct access to* $\mathbb{V}^0_{\varphi}(w)$.

Without any loss of generality, assume entities $i$,$j$ such as $i = j + 1$.

If entity $i$ has direct access to the sensor, e.g. the sensor physically triggers an alarm when

$\mathbb{V}^0_\varphi(w)$ returns *false,* and doubts the reading reported by $j$, then $\varphi$ is not MSDND(ES) secure with respect to $i$. But first the entity $i$ must doubt the reading enough to check the physical alarm.

*Proof.*

| | | |
|---|---|---|
| 1) | $\sim\varphi$ | The centrifuge speed is not optimal |
| 2) | $I_{i,0}\sim\varphi$ | Sensor turns on a physical alarm |
| 3) | $I_{1,0}\sim\varphi$ | Sensor reports $\sim\varphi$ electronically |
| 4) | $\vdots$ | The report is passed up the line |
| 5) | $I_{i,j}\varphi$ | $j$ electronically reports $\varphi$ |
| 6) | $\sim B_i I_{i,j}\varphi \vee \sim T_{i,j}\varphi$ | $i$ either mistrusts, or does not believe $j$ |
| 7) | $\sim(B_i I_{i,j}\varphi \wedge T_{i,j}\varphi)$ | DeMorgan's |
| 8) | $\sim B_i\varphi$ | $i$ does not believe $\varphi$ by rule C1 |
| 9) | $B_i I_{i,0}\sim\varphi$ | $i$ believes the sensor alarm is on |
| 10) | $T_{i,0}\sim\varphi$ | $i$ trusts the alarm |
| 11) | $B_i I_{i,0}\sim\varphi \wedge T_{i,0}\sim\varphi$ | Conjunction |
| 12) | $B_i\sim\varphi$ | by rule C1 |

□

In short, if there is a physical alarm on the centrifuge and $i$ looks because he or she does not trust the electronic reports, the optimal speed of the centrifuge is *not* MSDND(ES) secure with respect to any entity that bothers to check the physical alarm.

**Theorem 18.** *The speed of the centrifuge is MSDND(ES) secure for $SD^2$ during the attack phase for infected systems therefore $i$ will believe all is well or $\varphi$.*

*Proof.* By definition ($\varphi$ **xor** $\sim\varphi$) = *true,* so the first condition for MSDND(ES) is met. If $\varphi$ cannot be correctly evaluated in $SD^2$, then both conditions are met.

**Case 18.1.** *Centrifuge speed is nominal and $\varphi$ = $true$.*

| | | |
|---|---|---|
| 1) | $\varphi$ | Centrifuge nominal |
| 2) | $w \vDash \mathbb{V}^0_\varphi(w) = true$ | Definition of $w \vDash \mathbb{V}^0_\varphi(w)$ |
| 3) | $I_{1,0}\varphi$ | Sensor reports to virus |
| 4) | $B_1 I_{1,0}\varphi$ | Virus believes sensor report |
| 5) | $T_{1,0}\varphi$ | Virus trusts the sensors |
| 6) | $B_1 I_{1,0}\varphi \wedge T_{1,0}\varphi \rightarrow B_1\varphi$ | Axiom C1, Virus believes status |
| 7) | $I_{2,1}\varphi$ | **Virus *always* reports "all is fine"** |
| 8) | $B_2 I_{2,1}\varphi$ | PLC believes interface report |
| 9) | $T_{2,1}\varphi$ | PLC trusts reports |
| 10) | $B_2 I_{2,1}\varphi \wedge T_{2,1}\varphi \rightarrow B_2\varphi$ | Axiom C1 PLC believes all is well |
| 11) | $w \vDash \mathbb{V}^2_\varphi(w) = true$ | $\mathbb{V}^2_\varphi(w)$ **always** returns *true* |

**Case 18.2.** *Centrifuge speed is not nominal and* $\sim\varphi = true$.

| | | |
|---|---|---|
| 1) | $\sim\varphi$ | Centrifuge speed is not nominal |
| 2) | $w \vDash \mathbb{V}^0_\varphi(w) = false$ | Definition of $w \vDash \mathbb{V}^0_\varphi(w)$ |
| 3) | $I_{1,0}\sim\varphi$ | Sensor reports problem to virus |
| 4) | $B_1 I_{1,0}\sim\varphi$ | Virus believes sensor report |
| 5) | $T_{1,0}\sim\varphi$ | Virus trusts the sensors |
| 6) | $B_1 I_{1,0}\sim\varphi \wedge T_{1,0}\sim\varphi \rightarrow B_1\sim\varphi$ | Axiom C1, Virus believes status |
| 7) | $I_{2,1}\varphi$ | **Virus *always* reports "all is fine"** |
| 8) | $B_2 I_{2,1}\varphi$ | PLC believes interface report |
| 9) | $T_{2,1}\varphi$ | PLC trusts reports |
| 10) | $B_2 I_{2,1}\varphi \wedge T_{2,1}\varphi \rightarrow B_2\varphi$ | Axiom C1 PLC believes all is well |
| 11) | $w \vDash \mathbb{V}^2_\varphi(w) = true$ | $\mathbb{V}^2_\varphi(w)$ **always** returns *true* |

Since $T_{2,1}\varphi \wedge B_2 I_{2,1}\varphi \rightarrow B_2\varphi$, the PLC believes the lie told in step 7 in all cases. Therefore, unknown to entities in $SD^2$, $\mathbb{V}^2_{\varphi}(w)$ and $\mathbb{V}^2_{\sim\varphi}(w)$ cannot be evaluated. We now have all the requirements to conclude that $\varphi$ is MSDND(ES) secure from $SD^2$. □

During the attack phase, Stuxnet receives sensor reports and always reports to the PLC that the centrifuge is within acceptable operational parameters. Stuxnet has hijacked the interface between the sensor and the PLC.

It should be noted that the doxastic proof above has at its heart a violation of trust in the system. Briefly, line 2 states that the speed sensor on the centrifuge reports correctly that it is outside nominal operating speed. The virus has inserted itself into the buffer between the sensor and the PLC and hijacked the interface. Because this interface is designed to receive secure messages from the sensors on the centrifuge itself, the PLC trusts the reading (line 2) as if it came directly from the sensor. The virus *always* reports the centrifuge is operating normally, $\varphi$, to the PLC (line 6) whether $\varphi$ or ~$\varphi$.

The PLC trusts and believes all reports from the centrifuge. Stuxnet is a liar because it reports "all is well" even when it receives reports that the speed is not correct. The interface has been successfully hijacked by the virus and the PLC has no way to know that the virus is an intentional liar. Therefore the virus has successfully created a situation that is MSDND(ES) and the PLC does not take corrective action.

The sensor at $SD^0$ is able to correctly evaluate the situation. Should the sensor be directly connected to an alarm circuit or light that is seen by the human operator, the physical alarm Useless uranium will be produced until the centrifuge finally fails.

**Theorem 19.** *The speed of the centrifuge is MSDND(ES) secure for $SD^3$ during the attack phase for infected systems, therefore entities in $SD^3$ will trust that $\varphi = true$.*

During the attack phase, the system always reports the centrifuge is within acceptable operational parameters.

*Proof.*

By definition ($\varphi$ **xor** ~$\varphi$) = *true*, so the first condition for MSDND(ES) is met. If $\varphi$ cannot

be correctly evaluated in $SD^3$, then both conditions are met. The valuations $\mathbb{V}^3_{\varphi}$ and $\mathbb{V}^3_{\sim\varphi}$ both must rely upon reports from $SD^2$, therefore if $SD^2$ is MSDND(ES) secure $SD^3$ must also be MSDND(ES) secure. Belief and trust are also carried up from $SD^2$:

| | | |
|---|---|---|
| 1) | $\varphi \vee \sim\varphi \rightarrow B_2\varphi$ | Theorem 6.6.1 |
| 2) | $I_{3,2}\varphi$ | No problems reported to monitor |
| 3) | $B_3 I_{3,2}\varphi$ | Monitor believes interface report |
| 4) | $T_{3,2}\varphi$ | Monitor trusts all reports from 2 |
| 5) | $B_3 I_{3,2}\varphi \wedge T_{3,2}\varphi \rightarrow B_3\varphi$ | Monitor believes all is well |

Since $T_{2,1}\varphi \wedge B_2 I_{2,1}\varphi \rightarrow B_2\varphi$, the PLC believes the lie told in step 7 in all cases. Therefore, unknown to entities in $SD^2$, $\mathbb{V}^2_{\varphi}(w)$ and $\mathbb{V}^2_{\sim\varphi}(w)$ cannot be evaluated. We now have all the requirements to conclude that $\varphi$ is MSDND(ES) secure from $SD^2$. □

**Theorem 20.** *The speed of the centrifuge is MSDND(ES) secure for $SD^4$ and $SD^5$ during the attack phase for infected systems.*

The proof follows the same pattern as the proof of Theorem 6.6.1 and will not be given here.

## 6.7. REMARKS

MSDND(ES) and BIT logic can be used to model Stuxnet type attacks. Such attacks rely on MSDND(ES) and the inherent trust placed in the components of CPS to hide critical information from electronic monitoring and from human operators. Others [14] have discussed how difficult it is to thwart specifically targeted attacks such as APTs. Because Stuxnet-like attacks do not make an effort to steal information, there is no need for the virus to connect with the INTERNET. Therefore, monitoring out-bound traffic does not help.

Because such an attack replays valid readings, any effort to find problems through internal inconsistencies is also doomed. It is not feasible to eliminate the human components in large scale operations of CPS; therefore APT attacks will often be successful via social engineering. Once such a virus is in place, detection is complicated by human trust in

electronic systems. If we expect the electronic monitoring to give us correct results, a low-level attack on the physical sensor-monitor communications such as Stuxnet will succeed.

The importance of Corollary 6.6.1 is clear. All CPS must also have physical monitoring that can be used to verify the operation of the electronic monitoring or the next Stuxnet type attack will also succeed. Verifying cyber security with low level physical monitoring can break the role of trust in MSDND attacks. In the case of Stuxnet, the simple addition of a physical read out of the actual speed of the centrifuge would have broken the attack model if the human operator distrusted the cyber monitoring enough to verify the readings on the monitor.

Stuxnet type attacks can be broken. Consider the centrifuge system in light of Corollaries 6.6.1 and 6.6.1. If the centrifuge is equipped with a *physical* speedometer in addition to the cyber monitoring, the speedometer can be made to trip an audible alarm or a cyber alarm with physical diversity from the normal monitoring/control system. For example, the speedometer might close a hard-wired circuit to turn on a siren and flashing red light. This is equivalent to all entities having direct access to the valuation function $\mathbb{V}^0_\varphi(w)$. If this is true, then Theorem 6.6.1 holds and MSDND(ES) based attacks fail.

# 7. PHYSICAL ATTESTATIONS, NONDEDUCIBILITY, AND THE SMART GRID

Events in Cyber Physical Systems affect the cyber system, the physical system, and the interfaces shared by both and must be examined from all three perspectives. In the proposed electrical smart grid, an agent might lie about its power generation in the physical world by falsifying meter readings in such a way that the lie is not deducible by purely cyber means. This paper will show how the cyber system can use information from the physical system to break the nondeducible nature of the attack and reveal its source. It is simple to use physical power readings to detect many attacks, but if one of the goals is to preserve the privacy of the meter readings of each home, it is not a simple matter to determine the source of malicious attacks. Multiple Security Domains Nondeducibility will be used to model the cyber physical system information flows to describe the detailed nature of the attack. Physical invariants can be then be used as an attestation of the true state of the system to expose the malicious agent.

## 7.1. INTRODUCTION

Typical CPS consist of an observable physical system with an embedded cyber control system. One of the issues when examining such a system is viewing the CPS as a cyber system and a separate physical system. The interface between the cyber and physical systems complicates the overall system. Because information is coupled between the two systems, some unified approach must be taken to consider the cyber, physical, and interfaces when analyzing the CPS. If the analysis looks at the cyber system and ignores the physical system or looks only at the physical system, many malicious attacks will be missed. It is quite likely that the cyber system will demonstrate correct behavior while the physical system does not [19]. The framework for the electrical smart grid and the invariant analysis is published in Roth [19]. However, this work formalizes the analysis and extends the previous work by at least 70% [3].

The trust placed in the security of CPS plays a distinct role in an attack on the CPS. Many attacks on CPS rely heavily upon the cyber, physical, and human agents trusting completely the cyber side of the system and especially any cyber monitors [16]. Often these attacks can be defeated *if* the agents involved, either human or cyber, simply examine the physical layer. Any attack on a CPS must hide from detection by both the cyber layer and the physical layer, but many algorithms to secure CPS ignore one or the other. One way an attacker can hide is by using the concept of Nondeducibility against the CPS security. An attack is Nondeducible even if the attack is detected *if* the identity of the attacker cannot be correctly determined or deduced. In a case such as this, the attack may be discovered, but the attacker may successfully remain hidden behind information flow nondeducibility.

One of the most studied CPS of the future is the smart electrical grid which is proposed to use distributed cyber intelligence to manage the local production and consumption of energy in a small residential distribution system connected to a larger utility grid. Currently, the control of power generation and distribution rests solely with a trusted electrical utility. An electrical utility has no reason to be concerned over malicious reporting of false power generation as only the utility has significant electrical generation capacity. Indeed, the utility is concerned only with metering power consumption. Local generation of power, such as solar panels, is changing the established infrastructure as will sharing that generation with neighbors rather than selling power back to the utility. As power generation shifts more and more to non-utility owned sources, generation and consumption metering is no longer in the hands of the trusted utility and new malicious behavior is possible. The coming smart grid must be able to detect these behaviors and guard against them. This paper will look at the smart grid as envisioned by the FREEDM project [18] and use the same system framework and physical invariant as outlined and studied by Roth [19]. This work extends [19] by 70% by formalizing the analysis using MSDND and additional invariants to break nondeducibility in order to identify the malicious node while preserving the privacy of the remaining nodes.

In the current smart grid literature, one of the commonly studied attack scenarios is the fake data injection attack, where a malicious adversary compromises one or more intelligent meters to report an incorrect state of the local distribution network [53] [54]. McLaughlin demonstrated the vulnerability of current smart meters to such an attack [55]. Most research into false data injection attacks has been based upon the assumption that the attacker could not create an attack vector that the cyber control algorithm could not detect. However, even if an attack can be detected, it may still be unidentifiable in the sense that the system state cannot be correctly determined [56] [30]. If the system state cannot be correctly determined, the identity of the attack may still be unknown. This work explores an approach that validates the results from the cyber control system by examining the state of the physical system.

Attestation has long been used in cyber systems to test the correctness of processes by peer evaluations [57]. The same principles can be applied to the smart grid to help detect malicious processes [19]. The physical distribution lines in the grid act as a high integrity channel that can be viewed as broadcasting all activity to each smart meter. This can be used to dramatically increase the difficulty of hiding the source of a fake data injection attack. A CPS attack that exhibits intermittent malicious behavior is detectable when the physical and cyber systems are used to validate each other. What is more, because the cyber and physical actions are tightly coupled in a CPS, physical observations can often be mapped to uniquely identifiable cyber actions [58]. An attestation protocol can map physical observations to unique cyber actions to clearly identify the source of the false data injection.

The main contribution of this work will combine physical attestation, ND [8], and MSDND [2] to describe the role each plays in detecting a malicious agent in a local smart grid. It is simple to use physical measurements to determine that an attack has occurred, but if the attacker can obscure the source of the attack it may not be possible to correctly identify the source of the attack. A simple attack may easily betray the source, but it might very well be that a clever attack could be mounted so that results could have been produced by any one of a small set of nodes. In a case like this, the attack is deducible but the source

is nondeducible. If it is not possible to deduce with absolute certainty exactly which set of events has occurred, the events are nondeducible.

For example, if the physical measurements clearly point to an attack but not the identity of the attacking node, i.e. the attacker could be either node 1 or node 2, the attack is deducible but the source is not. Nondeducibility is normally looked at in terms of keeping information secure. A simple example of deducibility can be given. If a secret action occurs in a secure partition of a cyber system, the goal of nondeducibility is to insure that no unwanted side effects of the action allow someone without security access to the secure partition to deduce that a specific secret action happened. Suppose a newsman knew that the United States was planning a large military action on the other side of the world. If the newsman saw a pizza truck delivering a large order to the Pentagon, he might correctly deduce that an attack was eminent The secret timing of the attack would no longer be nondeducible. It takes only *one* event that leaks information to make the secret deducible.

This work will look at two types of nondeducibility, Sutherland's Nondeducibility and Multiple Security Domains Nondeducibility MSDND. Both types of Nondeducibility will be defined over all possible events in the system, or ES, using Kripke frames and modal methods to remove the requirement to analyze the trace of the system twice to determine nondeducibility. Furthermore, this work will point out the limitations of this approach in specific cases with the goal of aiding in the design of local smart grids that are more immune to this attack.

**7.1.1. Problem Statement.** Can nondeducibility models be used to study fake power injection attacks in the smart grid? How can physical attestation based upon the capabilities of existing power meters be used to break the Nondeducibility of fake power injection attacks [19]?

**7.1.2. The Organization of This Work.** A brief logical background of ND and MSDND is given in Section 3.3 along with a brief overview of Kripke [39] frames and models. In Section 7.2 the subset of a proposed smart grid is presented. While this work

discusses a subset of the smart grid, applying this work to an entire neighborhood smart grid is reasonable.

A nondeducible attack is presented in Section 7.3 and the key concepts and methods of physical attestation to make the attack deducible are presented in Section 7.4. An algorithm to break the attack is then presented in Section 7.5. Lastly, some concluding remarks are made.

## 7.2. SYSTEM OVERVIEW

Table 7.1. Nomenclature

| Symbol | Usage |
|---|---|
| $V_i$ | Voltage measured at point $i$ |
| $\delta_i, \theta_i$ | The phase angle between voltage and current at $i$ |
| $P_i$ | Actual power consumed/generated by house $i$ |
| $\hat{P}_i$ | Advertised power consumed/generated by house $i$ |
| $P_B$ | Actual power measured on the distribution bus |
| $\varepsilon$ | Small power variation at a node, it $\varepsilon$ is negligible for stable transmission. |
| $I_i$ | An Invariant of the system at physical location $i$ |
| $SD^i$ | Security Domain of node $i$ |
| $SD^B$ | Security Domain of the distribution bus (physical measurements) |
| $\varphi, \psi, \kappa$ | Any arbitrary logical expression |
| $\mathbb{V}^i_\varphi(w)$ | The valuation of $\varphi$ on world $w$ for entity $i$ |

This section presents an overview of the smart grid based upon the architecture developed by the FREEDM Systems Center [18] [59]. An attack scenario against this smart grid is then presented.

**7.2.1. The Smart Grid.** The smart grid consists of a number of neighboring houses on a single distribution line attached to an electric utility, see Figure 7.1. Each house is capable of variable electrical generation and has variable electrical consumption. Each house is equipped with a Solid State Transformer (SST) which, for the purposes of this work,

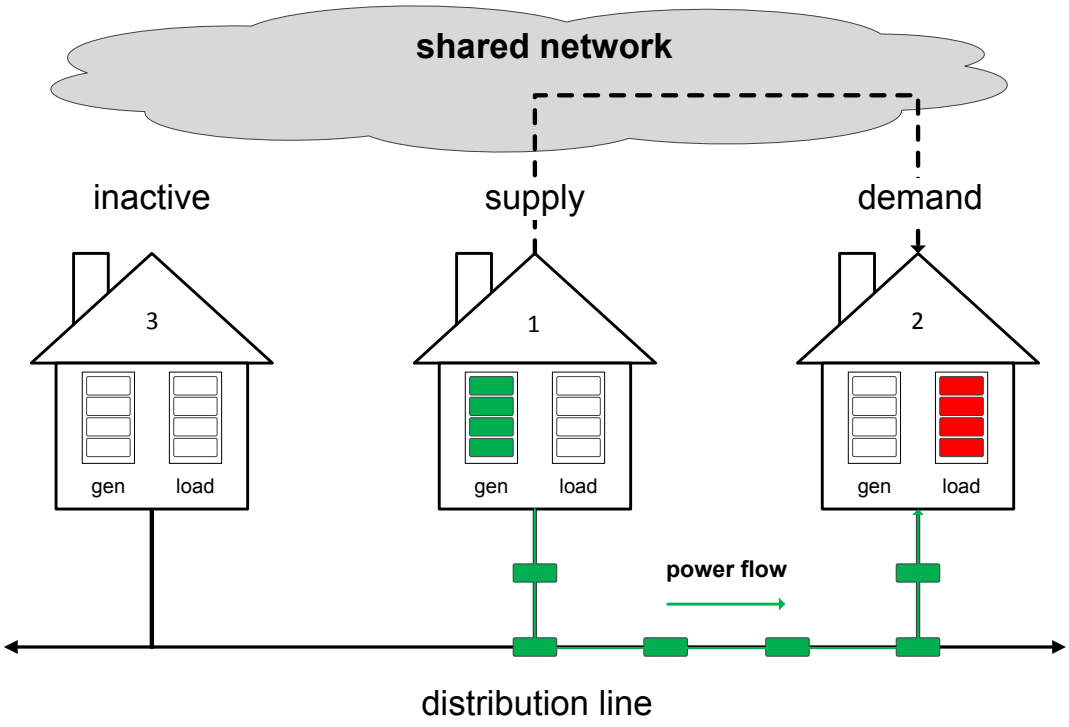



Figure 7.1. Smart Grid with Distribution Line

can be thought of a meter capable of measuring the voltage, current, and phase of the power entering and leaving the connection to the distribution line. These meters communicate with each other over a shared data network which is secured from the outside world.

Houses with excess generation capacity are said to be in a supply state while houses with more consumption than generation are said to be in a load state. Without loss of generality, houses in balanced generation and consumption will be ignored. Generation is assumed to be from some local storage (batteries) or a renewable energy source such as wind or solar power while load is assumed to be appliances within the house. The smart grid can draw additional power from the traditional electrical grid but this incurs additional costs. The preferred situation is for power to be produced and consumed locally via power transfers from house to house.

A house may supply its own load with its own generation or may pull power from the shared distribution line to satisfy the load. Any house may also push excess generated power

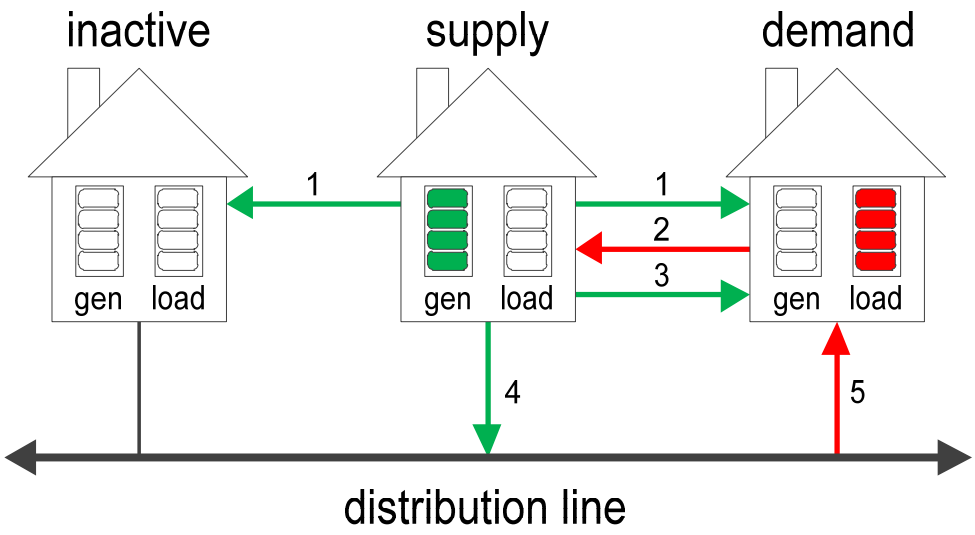



Figure 7.2. A Simple Power Migration in the Neighborhood

onto the distribution line to satisfy the load from other houses. The exchange of power by push/pull between houses is controlled by a distributed cyber intelligence that is embedded in the power controllers at each house. Push/pull is governed by power migration contracts negotiated between controllers and will not happen until these contracts have been formed. Power is migrated between houses based upon these contracts in the amounts required.

A migration is a sequence of steps, see Table 7.2, performed between two, and only two, houses on the same distribution line which is very similar to a two stage commit. A house with excess generation broadcasts a cyber message to all houses advertising the excess. Houses wishing to use excess power, i.e. in the demand mode, reply to the message. The supply house then selects one demand house and sends a message proposing a power migration contract. The supply house then increases generations and pushes power onto the distribution line. The demand house can then connect additional load. This results in a natural flow of power from the supply house to the demand house. However, these steps do not form an atomic transfer of power nor can the physical power be “signed” in any way. Both houses should either commit to the contract or both houses must abort the contract.

Table 7.2. Power Migration Steps

| | |
|---|---|
| 1. | Supply house ***advertises*** excess generation |
| 2. | All demand houses ***request*** power from supplier |
| 3a. | Supply house ***selects*** one demand house |
| 3b. | Supply house ***increases*** its local generation |
| 4. | Selected demand house ***increases*** its local load |

Table 7.3. Malicious House Power Migration

| | |
|---|---|
| 1. | Malicious house ***advertises*** its excess generation |
| 2. | All demand houses ***requests*** power from malicious supplier |
| 3a. | Malicious supply house ***selects*** one demand house |
| 3b. | Malicious supply house ***does not increase*** its local generation |
| 4. | Selected demand house ***increases*** its local load |

If not, the increased load will lead to purchasing power from the more expensive electric utility.

**7.2.2. Fake Power Injection Attacks.** One of the most commonly studied smart grid attack vectors is a fake power injection attack [53] [55], see Figure 7.3. A malicious house must first compromise the controller that connects it to the distribution line. It then follows all the cyber requirements of a legitimate power migration as in Table 7.2 without ever pushing any power to the grid. Because the demand house has already increased the load on the distribution system one of three *bad* outcomes must occur, either the voltage on the distribution system will decrease leading to instability and possible failure, the demand house must again decrease its load, or the smart grid must purchase more expensive power from the electrical utility. None of these outcomes is desirable and are exactly the situations the smart grid is designed to minimize.

The malicious house broadcasts a cyber advertisement to all houses that it has excess power. Those houses in demand mode send back a reply. The malicious house then selects

one of the demand houses and forms a migration contract. However, the malicious house *does not increase its generation and does not push power onto the distribution line.* In short, the malicious house never completes step 3b of Table 7.2. The unsuspecting demand house then connects extra internal load and begins drawing power from the distribution line. Either the distribution line will become unbalanced or some other generation source must supply additional power [58]. As no other house is likely to be in supply mode, the increased power will be supplied by the commercial utility and the local smart grid will have to pay for the power. In effect, the malicious house has completed the power migration contract in the cyber realm but no physical power has been transfered.

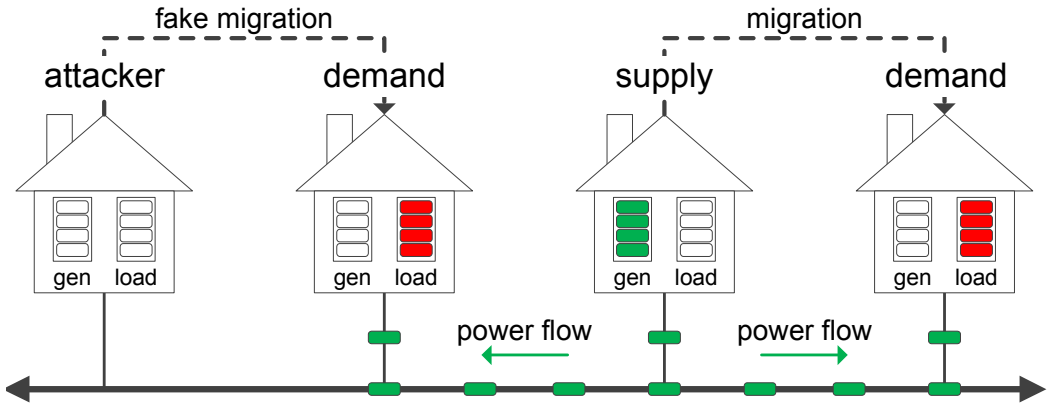



Figure 7.3. A Fake Power Injection Attack

How can this happen in reality? The issue lies in steps 3a and 3b of Table 7.2. It is not possible to combine these two actions into one atomic step and the malicious house can make use of this fact. Because power cannot be "signed", the demand house cannot distinguish a legitimate *select* message which is followed by power generation from a malicious *select* message which will not be followed by power generation. In this case,

the malicious house has successfully broken from the power migration protocol. The two phase commit has not been completed and a power imbalance has been produced.

If an attack of this form is carried out, there are two possible results. If the attack can be detected, legitimate demand houses can reduce the amount of power being pulled from the distribution line by turning off appliances (load shedding) in spite of existing legitimate power migration contracts. If the attack is not detected, the system will operate in an unbalanced state that is further from the optimal state. If repeated attacks are made, this will most likely result in the system becoming completely unstable. This instability could easily result in a blackout of the smart grid distribution system if additional power is not purchased from the main grid.

## 7.3. NONDEDUCIBLE ATTACK

**7.3.1. Formal System Model.** Without loss of generality, the smart grid is assumed to have a bus structure as in Figure 7.1. Houses in direct communication form groupings. Ideally, these grouping or segments of the smart grid will comprise all the houses on a common distribution bus but this may not always be the case. Houses with balanced generation/consumption can safely be ignored as well as the larger utility grid. Houses are labeled $i = 1, 2, \ldots, n$ and share a common shared media communications network where all houses *could* easily monitor all messages. All houses are connected by a common power distribution which all houses may measure. However, the meter for each house is private and may not be read by any other house.

The detailed messages and actions for a power migration are given in Table 7.4 and a sample power migration is shown in Figures 7.2 and Table 7.5. In this procedure as part of a CPS, steps 1 through 3 are purely cyber messages that do not affect the physical power distribution; however, steps 4 and 5 are purely physical and cannot be seen in the cyber portion of the system. This uncoupling of the CPS into purely cyber actions and purely physical actions is key to the success of the fake power injection attack.

Table 7.4. Power Migration Messages and Actions

| | |
|---|---|
| $adv(i)$ | Advertisement of excess power |
| $increase(i, x)$ | Increase in power generation of $x$ |
| $load(i, x)$ | Load at $house_i$ changed |
| $request(i, x)$ | Request for power migration |
| $select(i, j, x)$ | Offer of migration contract |
| $end()$ | end all algorithms |

Table 7.5. Good Power Migration from 1 to 2

| | | |
|---|---|---|
| 1. | $adv(1)$ | $house_1$ ***advertises*** excess power |
| 2. | $request(2, 5)$ | $house_2$ ***requests*** 5 units of power |
| 3. | $select(1, 2, 4)$ | A migration of 4 units is offered |
| 4. | $increase(1, 4)$ | $house_1$ ***increases*** generation by 4 |
| 5. | $load(2, 4)$ | $house_2$ ***increases*** its usage by 4 |

Table 7.6. Bad Power Migration from 1,3 to 2

| | | |
|---|---|---|
| 1. | $adv(1)$ | $house_1$ ***advertises*** excess power |
| 1a. | $adv(3)$ | $house_3$ ***advertises*** excess power |
| 2. | $request(2, 10)$ | $house_2$ ***requests*** 10 units of power |
| 3. | $select(1, 2, 4)$ | A migration of 4 units is offered |
| 3a. | $select(3, 2, 4)$ | A migration of 4 units is offered |
| 4. | $increase(1, 4)$ | $house_1$ ***increases*** generation by 4 |
| 5. | $load(2, 8)$ | $house_2$ ***increases*** its usage by 8 |

**7.3.2. Attack Analysis.** An adversary may exploit the power migration to allow billing of power migrations that were never completed. Assume houses 1 and 2 are honest and house 3 is a malicious house. Consider the sequence of events in Table 7.6, keeping in mind that all messages *could* be monitored by any house. The adversary mimics the actions of $house_1$ except $house_3$ does not increase generation as agreed to in the power migration. The distribution power is less than the increased demand. Either $house_3$ must immediately decrease the load on the system or some other source must generate 4 more units of power.

This is a bad outcome. One possible solution is to purchase power from the utility. But the question remains, can the smart grid members determine which house, 1 or 3, failed to fulfill the power migration?

**Constraint** (Privacy Constraint)**.** Due to privacy considerations, a meter, $meter_i$, may be read only by the house to which it is attached. That is, $\exists \mathbb{V}^i_{P_j}(w)$ if, and only if, $i = j$.

**Theorem 21.** *A* **fake power injection attack** *in a three node segment meets the requirements for MSDND(ES) from the view of the messaging system (cyber).*

This is a problem when an attack occurs. It is possible to determine an attack has occurred, but the proof will show that it is *not* possible to determine the identity of the attacker. The attacker may perform this attack successfully at will.

*Proof.* The messaging system can be monitored by all houses. The sequence of cyber messages, $\varphi$, is the same regardless of which house is dishonest. True, the messages have a particular sequence but in a distributed system establishing causality based upon the time sequence of messages is rarely reliable. The cyber partition of the system cannot determine the attacker. Therefore, the only relevant events must be after step 3 in Table 7.6.

It is simple to measure the power on the bus and see that the change is given by $P'_B = P_B + 4$ where $P'_B$ is the power on the bus during the physical power migration. The actions by the three houses must yield this result.

**Case 1:** Neither $house_1$ nor $house_3$ performed step 4 $increase(1, 4)$ or $increase(3, 4)$.
In this case, $P'_B = P_B$ which is not what is observed. This case did not occur.

**Case 2:** Both $house_1$ and $house_3$ performed step 4 $increase(1, 4) \wedge increase(2, 4)$.
In this case, $P'_B = P_B + 8$ which is not what is observed. This case did not occur.

**Case 3:** $house_1$ performed step 4 $increase(1, 4)$.
In this case, $P'_B = P_B + 4$ which is what is observed. Let this case be denoted as $w_1$. Then in $w_1$, the cyber steps 1 through 3 of the migration lead to the set of messages denoted by $\varphi$.

**Case 4:** $house_3$ performed step 4 $increase(3,4)$.

In this case, $P'_B = P_B + 4$ which is what is observed. Let this case be denoted as $w_3$. Then in $w_3$, the cyber steps 1 through 3 of the migration lead to the set of messages denoted by $\varphi$.

Cases 1 and 2 do not match the power measurable on the physical distribution system.

Let:

$\varphi_1$ be "$increase(1,4)$ occurred."

$\varphi_3$ be "$increase(3,4)$ occurred."

$\mathbb{V}^C_{\varphi_1}(w)$ be a valuation function for the messaging system for $\varphi_1$

$\mathbb{V}^C_{\varphi_3}(w)$ be a valuation function for the messaging system for $\varphi_3$

| | | |
|---|---|---|
| 1. | $\varphi_1 \vee \varphi_3$ | Case 1 |
| 2. | $\sim(\varphi_1 \wedge \varphi_3)$ | Case 2 |
| 3. | $\varphi_1$ **xor** $\varphi_3$ | 1, 2 Conjunction |
| 4. | $\nexists\, \mathbb{V}^i_{p_j}(w) : i \neq j$ | privacy constraint |
| 5. | $\nexists\, \mathbb{V}^C_{\varphi_1}(w)$ | privacy constraint |
| 6. | $\nexists\, \mathbb{V}^C_{\varphi_3}(w)$ | privacy constraint |
| 7. | $\{[\nexists\, \mathbb{V}^C_{\varphi_1}(w)] \wedge [\nexists\, \mathbb{V}^C_{\varphi_3}(w)]\}$ | 5 and 6 |
| 8. | The attack is MSDND(ES) | statements 3 and 7 |

Steps 3 and 7 are the clauses needed to show MSDND(ES) for $\varphi_1$ and $\varphi_3$ in the cyber security domain, that is:

$\exists w \in W : w \vdash \Box[\varphi_1$ **xor** $\varphi_3]$

$\wedge [w \vDash (\nexists\, \mathbb{V}^C_{\varphi_1}(w) \wedge \nexists\, \mathbb{V}^C_{\varphi_3}(w))]$ □

One outcome of Theorem 21, is that it is not possible to determine if the system state is such that $w = w_1$ or $w = w_3$ because it is consistent with both. The proof is obvious from

Theorem 21 because case 1 and case 2 are not possible if $P'_B = P_B + 4$.

**Theorem 22.** *A* **fake power injection attack** *on a three node segment meets the requirements for MSDND(ES) from the view of the distribution line (physical).*

*Proof.*

The proof follows the same reasoning as Theorem 21 to show:

$$\varphi_1 \textbf{ xor } \varphi_3 \wedge \left\{ \left[ \nexists\, \mathbb{V}^P_{\varphi_1}(w) \right] \wedge \left[ \nexists\, \mathbb{V}^P_{\varphi_3}(w) \right] \right\}.$$

It follows that the attack is MSDND(ES) secure from the purely physical viewpoint. □

**Corollary 3.** *Measurement of the power transferred on the distribution line leaks information about the CPS, specifically: exactly one house increased generation.*

The obvious conclusion from Theorem 21 and Theorem 22 is that the **fake power injection attack** is nondeducible and will succeed under these conditions. The ability to break the Nondeducibility of the attack does not exist when measuring only the power on the bus and at the supply node. More information is needed to break the attack.

One possible solution is to deny multiple concurrent power migrations to take place on any segment of the smart grid. Since only a single *increase* command would be allowed to follow each *select* command, the system would require a handshake to occur and any fake power injection attack would become deducible. This is the corrective approach suggested by the paper that introduced the attack [30]. However, Roth [19] suggested using the physical properties of the CPS and the inherently leaked information provided by monitoring the various voltages on the shared power bus. Thus an apparent weakness of

CPS, the leakage of information from direct physical observations, can be used to enhance the functions of the system using a technique introduced in the same paper, "physical attestation".

**7.3.3. The Role of Trust in the Attack.** One of the keys to the success of the **fake power injection attack** is the trust inherent in the system. Each agent in the system inherently trusts every other agent without taking into account that an agent may be a liar.

*Definition* 1 (Liar [9]).

- Agent $i$ is an intentional liar if $U_i\varphi \wedge B_i{\sim}\varphi$
- Agent $i$ is an irresponsible liar if $U_i\varphi \wedge {\sim}B_i\varphi$
- An intentional liar is also an irresponsible liar

## 7.4. PHYSICAL ATTESTATION AND CONSERVATION OF ENERGY

Any single node of the smart grid does not possess enough information to break a fake power injection attack nor can enough information be gathered by measuring the power on the common bus. An aware attacker can easily modify the actions at the malicious node to hide from detection. It is possible to devise a distributed algorithm to independently verify the power injected at a single node and foil this attack. The following sections will provide the requirements for physical attestation. The collected attestation is then used as input to an algorithm to correct the malicious values reported (faked meter readings and power generation). Using the corrected values, the smart grid segment will function correctly in the presence of a single attacker.

**7.4.1. Conservation of Energy and Kirchoff's Law.** In order to determine when a reported reading has been falsified, a set of invariants must be defined. The invariant will be true only when the reported values make sense and will evaluate to false when there is a malicious action by some node. No other values are possible for an invariant. Due to privacy considerations, only the house served by a meter can read the physical meter;

however, conservation of energy can be used to form an invariant. Consider a small segment of the smart grid forming the circuit in Figure 7.4. For such a circuit, the invariant $I_j$ must hold such that:

$$\{I_j : P_{ij} + P_j - P_{jk} = 0 \pm \varepsilon\}. \tag{7.1}$$

If the invariant $I_j$ does not hold, one of the nodes that contributes to the equation 7.1 must

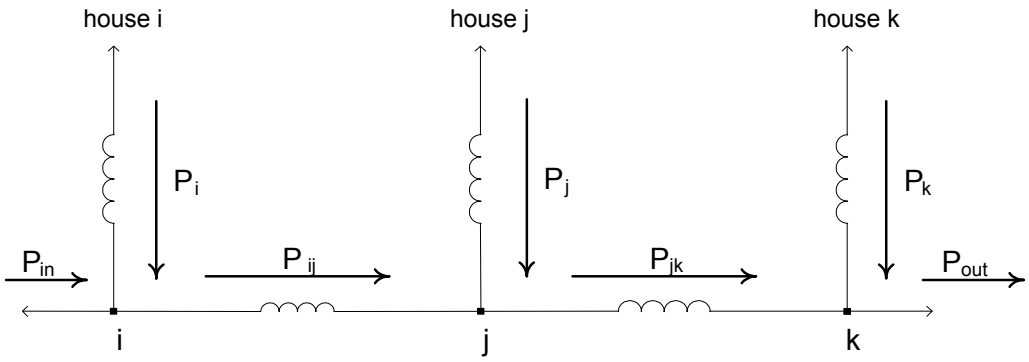
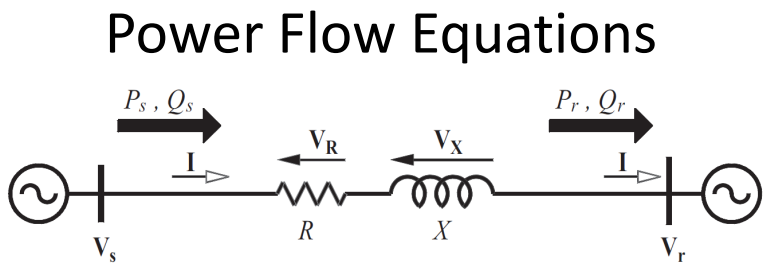



Figure 7.4. Power Attestation to Form Invariant

be malicious. One of the reported values is dishonest, but as was shown earlier it is not possible to use the values at one node to uncover the attacker.

Suppose node $j$ is to be checked by physical attestation. The reported generation $\hat{P}_j$ must somehow be compared to the actual generation $P_j$. But $P_j$ can only be directly measured at $house_j$. This is a privacy violation. To verify $\hat{P}_j$, the values for the other two power flows must be calculated. This can be done by using the reports from node $j$ and the neighboring node on either side using the line power equations, see Figure 7.5.

The values of $P_{ij}$ and $P_{jk}$ can be calculated from the voltages $V$ and phase angles $\theta$ as reported by the neighbors $i$ and $k$. The assumption is made that each node on the smart




$$P_s = \frac{V_s}{R^2 + X^2}\left[ R\left\{V_s - V_r \cos(\delta_s - \delta_r)\right\} + XV_r \sin(\delta_s - \delta_r)\right]$$

$$P_r = \frac{V_r}{R^2 + X^2}\left[ R\left\{V_s \cos(\delta_s - \delta_r) - V_r\right\} + XV_s \sin(\delta_s - \delta_r)\right]$$

Figure 7.5. Power Flow Calculations for a Segment Between Nodes

grid segment is equipped with the ability to measure and store the voltage and phase angle on the public side of its connection, for example as point $i$ on Figure 7.4. To allow these calculations to be performed, the voltages and phase angles must be measured by a device, such as a phasor measurement unit (PMU), and stored. At a later time, each house will report the history of voltage and phase angle to a "verifier" unit. Unfortunately, a malicious node would simply report false values for its measurements of voltage and phase angle. Given that a malicious node may report erroneous readings for voltage, phase angle, or generated power, the verifier must compare the reported values with calculated values to determine the truth value of the invariant.

**7.4.2. Three Node Attestation.** Cases where the malicious node violates invariants so as to immediately allow the verifier to use the information obtained via the DGI algorithm to determine the identity of the attacking node, can safely be ignored as being trivial. An example of such an attack would be if node $i$ lied about incoming power, $P_{in}$ and its current generation, $P_i$, so that they did *not* compensate, attack pattern $\varphi_1$ would be obvious

because only invariant $I_i$ would be violated. The implicit assumption is that the attacking node has knowledge of the distribution system and which invariants will be violated by misreporting data. In short, the malicious node will intelligently avoid detection, but the patterns $\varphi_1, \varphi_4, \varphi_5,$ and $\kappa_1$ reflect the invariants violated by a less than intelligent attack.

Using the data from one node and its two neighboring nodes does not break the Nondeducibility of a single malicious node performing a fake power injection attack [19]. In three node attestation as illustrated in Figure 7.4, there are two cases: a malicious node on either edge of the group of three nodes ($i$ or $k$), or the node in the center between the two on the edges ($j$).

**Theorem 23.** *In a three node attestation with one malicious node, any node could launch a fake power injection attack that is MSDND(ES) secure.*

The malicious node cannot be identified from the invariants using reported or calculated data.

*Proof.*

Regardless of which node reports false data, the invariant can be formed at three nodes $i, j,$ and $k$ on the shared power bus.

$$I_i = \quad P_{in} + P_i - P_{ij} = 0 \pm \varepsilon \tag{7.2}$$

$$I_j = \quad P_{ij} + P_j - P_{jk} = 0 \pm \varepsilon \tag{7.3}$$

$$I_k = \quad P_{jk} + P_k - P_{out} = 0 \pm \varepsilon. \tag{7.4}$$

**Remarks 1.** Notice $I_i$ and $I_k$ depend upon power flows *that cannot be independently verified.* These two power flows, $P_{in}$ and $P_{out}$ can only be reported by nodes $i$ and $k$ respectively. There are no nodes on either side that report data to the verifier.

**Case 23.1.** *Node $i$ is malicious.*

Table 7.7. Impact of One Malicious Node

| | Node | Falsified Values | Violated Invariants |
|---|---|---|---|
| $\varphi_1$ | $i$ | $P_{in}P_i$ | **none** |
| $\varphi_2$ | $i$ | $P_i$ | $I_i$ |
| $\varphi_3$ | $i$ | $V_i\theta_i$ | $I_iI_j$ |
| $\varphi_4$ | $i$ | $P_iV_i\theta_i$ | $I_j$ |
| $\varphi_5$ | $i$ | $P_{in}P_iV_i\theta_i$ | $I_j$ |
| $\psi_1$ | $j$ | $P_j$ | $I_j$ |
| $\psi_2$ | $j$ | $V_j\theta_j$ | $I_iI_jI_k$ |
| $\psi_3$ | $j$ | $P_jV_j\theta_j$ | $I_iI_k$ |
| $\kappa_1$ | $k$ | $P_{out}P_k$ | **none** |
| $\kappa_2$ | $k$ | $P_k$ | $I_k$ |
| $\kappa_3$ | $k$ | $V_k\theta_k$ | $I_jI_k$ |
| $\kappa_4$ | $k$ | $P_kV_k\theta_k$ | $I_j$ |
| $\kappa_5$ | $k$ | $P_{out}P_kV_k\theta_k$ | $I_j$ |

If node $i$ is malicious, there are five possible patterns of invariants from a fake power injection attack, see Table 7.7 with at least one pattern that is MSDND(ES), $\varphi_1$. Because there is no other node that can verify the value of $P_{in}$, an intelligent attacker would simply report $P_{in} = 0$ and $P_1 = P_{in}$. The attack would not violate any invariants and therefore would be undetectable. If an increase $P_B$ is measured and $i$ reports an increase in generation, then $P_{in}$ increased or $P_i$ increased but not both (from the measurement of $P_B$). No other node can verify the values of $P_{in}$ or $P_i$ so there are no valuations $\mathbb{V}_{P_{in}}(w)$ and $\mathbb{V}_{P_1}(w)$. Therefore we can form both the clauses for MSDND,

$$w \vdash [(P_{in} \textbf{ xor } P_1)] \wedge w \vDash [(\nexists\, \mathbb{V}_{P_{in}}(w)) \wedge (\nexists\, \mathbb{V}_{P_1}(w))]\,.$$

Therefore, if $i$ lies correctly about the readings, this case is MSDND(ES).

**Case 23.2.** *Node $j$ is malicious.*

If node $j$ reports false values for $P_jV_j\theta_j$, two invariants $I_j$ and $I_k$ will be violated which corresponds to $\psi_3 = true$. However, the same pattern of violations occurs when node $k$ lies about the values $V_k\theta_k$ which is $\kappa_3 = true$. Since we have only one malicious node, we

can show MSDND(ES) as follows:

1. $\psi_3$ **xor** $\kappa_3$ there is only one malicious node
2. $\nexists\, \mathbb{V}_{P_j}$ no one but $j$ can read $P_j$
3. $\therefore \nexists\, \mathbb{V}_{\psi_3}(w)$ privacy
4. $\therefore \nexists\, \mathbb{V}_{\kappa_3}(w)$ similar reasoning.

Therefore, an intelligent node $j$ can launch at least one attack that is MSDND(ES).

**Case 23.3.** *Node $k$ is malicious.*

Node $k$ can perform an attack that mirrors the attack given for node $i$. The proof of MSDND(ES) for this case follows the same reasoning as when node $i$ is malicious and will not be given here. Since there exists at least one MSDND(ES) attack for each of the three

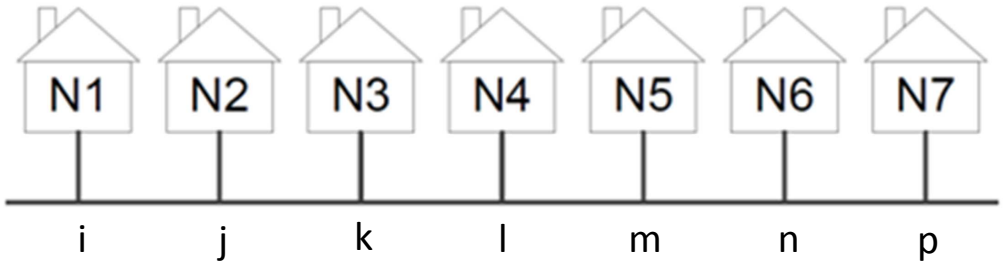



Figure 7.6. The Seven Node Attestation Framework

nodes, in a three node attestation with one malicious node any node could launch a fake power injection attack that is MSDND(ES) secure. □

**7.4.3. Attack Analysis for a Distribution Segment of Seven Nodes.** Without loss of generality, assume node $i$ is a supply node and node $j$ is a demand node. An independent verifier monitoring the DGI messages and voltage on the distribution bus could run Algorithm 1 to attest to the actual generation.

The seven node attestation model extends the region over which the invariants can be formed and independently verified. The issue lies with the edge nodes reporting the

Table 7.8. Seven Node Invariant Violation Patterns

| Pattern | Node | Falsified Values | Violated Invariants |
|---|---|---|---|
| $\varphi_1$ | $i$ | $P_{in}P_i$ | **none** |
| $\varphi_2$ | $i$ | $P_i$ | $I_i$ |
| $\varphi_3$ | $i$ | $V_i\theta_i$ | $I_iI_j$ |
| $\varphi_4$ | $i$ | $P_iV_i\theta_i$ | $I_j$ |
| $\varphi_5$ | $i$ | $P_{in}P_iV_i\theta_i$ | $I_j$ |
| $\psi_1$ | $j$ | $P_j$ | $I_j$ |
| $\psi_2$ | $j$ | $V_j\theta_j$ | $I_iI_jI_k$ |
| $\psi_3$ | $j$ | $P_jV_j\theta_j$ | $I_k$ |
| $\kappa_1$ | $k$ | $P_k$ | $I_k$ |
| $\kappa_2$ | $k$ | $V_k\theta_k$ | $I_jI_kI_\ell$ |
| $\kappa_3$ | $k$ | $P_kV_k\theta_k$ | $I_jI_\ell$ |
| $\lambda_1$ | $\ell$ | $P_\ell$ | $I_\ell$ |
| $\lambda_2$ | $\ell$ | $V_\ell\theta_\ell$ | $I_kI_\ell I_m$ |
| $\lambda_3$ | $\ell$ | $P_\ell V_\ell\theta_\ell$ | $I_kI_m$ |
| $\mu_1$ | $m$ | $P_m$ | $I_m$ |
| $\mu_2$ | $m$ | $V_m\theta_m$ | $I_\ell I_mI_n$ |
| $\mu_3$ | $m$ | $P_mV_m\theta_m$ | $I_\ell I_n$ |
| $\alpha_1$ | $n$ | $P_n$ | $I_n$ |
| $\alpha_2$ | $n$ | $V_n\theta_n$ | $I_mI_nI_p$ |
| $\alpha_3$ | $n$ | $P_nV_n\theta_n$ | $I_m$ |
| $\beta_1$ | $p$ | $P_{out}P_p$ | **none** |
| $\beta_2$ | $p$ | $P_p$ | $I_p$ |
| $\beta_3$ | $p$ | $V_p\theta_p$ | $I_nI_p$ |
| $\beta_4$ | $p$ | $P_pV_p\theta_p$ | $I_n$ |
| $\beta_5$ | $p$ | $P_{out}P_pV_p\theta_p$ | $I_n$ |

power flowing into and out of the segment that cannot be independently verified. As was shown with the three node framework, the values at the edge nodes, house N1 and N7, are MSDND(ES) and this makes attestation problematic for houses N2 and N5 as well. Examination of Table 7.8 shows that on a segment of seven nodes, any node may launch a fake power injection attack that is MSDND(ES) *except the center node.*

**Theorem 24.** *A* fake power injection attack *meets the requirements for MSDND(ES) from the view of the messaging system.*

The proof follows exactly the proof of Theorem 21.

**Theorem 25.** *A* fake power injection attack *meets the requirements for MSDND(ES) from the view of the distribution bus.*

Again, monitoring the distribution bus yields the same information as in the three node case and the proof follows exactly that of the three node segment.

**Theorem 26.** *In a seven node attestation with one malicious node, any node other than the center node could launch a fake power injection attack that is MSDND(ES) secure.*

*Proof.* The invariants can still be formed in the same manner as for the three node segment. It is trivial to determine if an attack has occurred because the cyber messages (DGI) show plainly that an increase should occur but monitoring the actual distribution bus power $P_B$ shows that no increase actually took place. However, the *source* of the attack is unknown and in most cases cannot be determined.

**Case 26.1.** *If one of the two end nodes is malicious (node $i$ or $p$), the node can launch an attack such that the source is MSDND(ES) secure.*

In this case the proof follows exactly the reasoning for the end nodes of the three node segment. There is no need to repeat the proof here.

**Case 26.2.** *Node $j$ can launch an attack such that the source is MSDND(ES) secure.*

Suppose node $j$ intentionally falsifies a reading of its generated power, $P_j$ to instigate a *fake power injection attack*. From Table 7.8, this would cause invariant $I_j$ to fail ($\psi_3$). However, a failure of $I_j$ can also be caused by node $i$ misreporting values $P_{in}, P_i, V_i,$ and $\theta_i$ or $\varphi_5$. Of course, node $i$ could also misreport only the values for $P_{in}, P_i, V_i,$ and $\theta_i$, or pattern $\varphi_4$, but that is essentially the same attack. Therefore, for this case we need to show that $\varphi_5$ and $\psi_1$ are MSDND(ES) secure.

1. $\varphi_5$ **xor** $\psi_1$ $I_j$ and $P_B$ increased
2. $\nVdash_{\varphi_5} (w)$ privacy
3. $\nVdash_{\psi_1} (w)$ privacy

Therefore, this case is MSDND(ES) secure: $w \vdash [(\varphi_5 \textbf{ xor } \psi_1)] \wedge w \vDash [(\nVdash_{\varphi_5} (w)) \wedge (\nVdash_{\psi_1} (w))]$.

**Case 26.3.** *Node $k$ can launch an attack such that the source is MSDND(ES) secure.*

The two patterns, $psi_3$ and $\kappa_1$ are obviously the same and MSDND(ES) secure by reasoning similar to node $j$.

**Case 26.4.** *The center node $\ell$ of the segment cannot launch an attack such that the source is MSDND(ES) secure.*

The patterns of invariant failure caused by node $\ell$ are unique. No attack launched by a different malicious node is the same as the pattern for this node. Therefore, the clauses required to show MSDND(ES) cannot be constructed. Given any contiguous set of seven nodes on a single distribution line, it is possible to determine if the center node is malicious.

**Case 26.5.** *Node $m$ can launch an attack such that the source is MSDND(ES) secure.*

The two patterns, $alpha_3$ and $\mu_1$ are obviously the same and MSDND(ES) secure by reasoning similar to node $j$.

**Case 26.6.** *Node $n$ can launch an attack that is MSDND(ES) secure.*

The proof is a mirror image of the proof for node $j$. □

## 7.5. FORCING DEDUCIBILITY

The results of previous theorems can be used to force deducibility upon some of the nodes of any arbitrary segment with more than six contiguous nodes on the smart grid. It is necessary to create an outside verifier with access to the DGI messages, see Table 7.9, and an independent measurement of the power on the distribution bus, see also Figure 7.5. By running the algorithm given in Algorithm 1, anytime a power migration contract is made the supply node can be verified if the contract is not completed within a predetermined time.

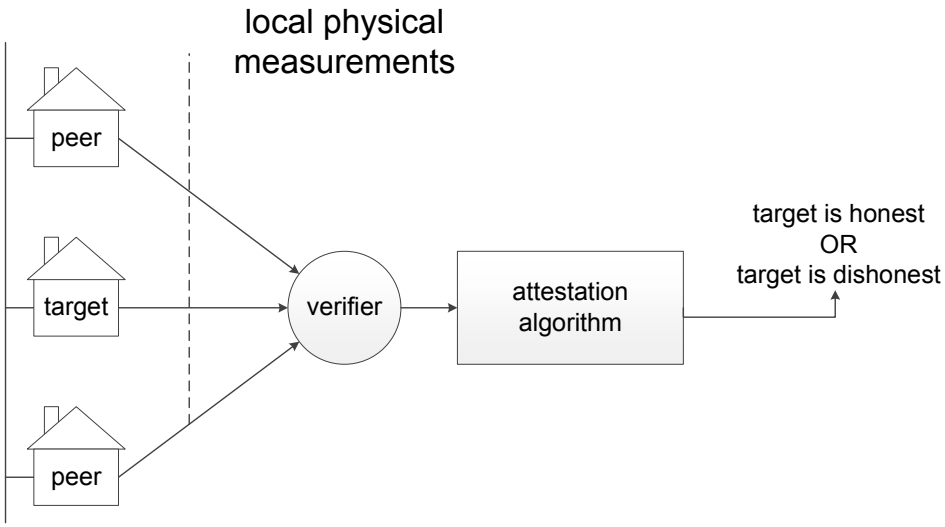



Figure 7.7. The Independent Verifier

Table 7.9. Message Types for Monitor Algorithm

| $mtype$ | Message | Purpose |
|---|---|---|
| a | advertise($i$) | Advertise supply status |
| r | request($i, x$) | Request power migration |
| s | select($i, j, x$) | Select node for power contract |
| e | end() | Gracefully end algorithm |

**Algorithm 10** Distributed Grid Intelligence Monitor Algorithm

```
1:  procedure DGI MONITOR
2:      var
3:      mtype : character := NULL;                          ▷ See Table 7.9
4:      msg : message := NULL;
5:      n : integer := N;                                   ▷ N is the number of nodes
6:      i, j : integer :=0,0;
7:      P_B, x : real :=0.0, 0.0;
8:      C[n][n] : real := 0.0;                              ▷ Power Migration Contract
9:      RUN : boolean :=true;
10:     NOP : empty statement, no operation;
11:     begin
12:         while RUN do
13:             Read P_B;
14:             wait(msg);
15:             Read mtype, msg;
16:             if
17:                 [] mtype == a → NOP;
18:                 [] mtype == r → NOP;
19:                 [] mtype == s → DGIsel(i, j, x);
20:                 [] mtype == e → RUN := false;
21:             fi;
22:         end while
23:     end;
24: end procedure
```

**7.5.1. Comments on the Algorithms.** Some explanation of the proposed algorithms to be run on an independent verifier is in order. Technically, the algorithms are not part of the FREEDM DGI, but are run on a verifier that can monitor the DGI messages and can perform independent and direct measurements of the power state of the distribution bus at the point where each house connects to the electrical smart grid. Measurements on the public side of the house meters do not violate the privacy of the house owner. Algorithm 1, or DGI Monitor, monitors the message traffic in the DGI and spawns an independent instance of Algorithm 2 DGISEL, to watch for a node to complete a migration contract. It does this by periodically performing a secure power calculation[11], Algorithm 3 or DGISPC, of the expected power generation by the supply node. If this calculation does not show a

[11]This calculation could overload the verifier, so provisions are made to control how often this check is done by using a system variable, $tick$, to determine the time interval between checks.

power migration by the supply node within a system-wide predetermined timeout, an alert is generated.

**Algorithm 11** Distributed Grid Intelligence SELECT Algorithm

```
1:  procedure DGISEL(integer i, integer j, real x)
2:      var
3:      ε : real := VARIANCE;                          ▷ system wide value
4:      timeout : real :=TIMEOUT;                      ▷ system wide value
5:      tick : real :=TICK;                            ▷ system wide value
6:      time, t : real :=0,TIME;                       ▷ system wide value
7:      P_B, P_i, P_0, P_cal : real :=0.0,0.0,0.0, 0.0;
8:      passed : boolean :=false;
9:      i, j : integer;

10:     begin
11:         Read P_B;
12:         P_0 := P_B;
13:         while ((time < timeout) ∧ (!passed) do
14:             get P_i for time t from node i;
15:             P_cal := DGISPC(i, t, ε)
16:             if (P_i ± ε) == P_cal then passed :=true;
17:             end if
18:             wait(tick);
19:             time := time + tick;
20:             Read P_B;
21:         end while
22:         if !passed then
23:             ALERT("NODE i FAILED ATTESTATION @ TIME t");
24:             P_i = P_cal
25:             ALERT("POWER P_i CORRECTED");
26:         end if
27:     end;
28: end procedure
```

**Algorithm 12** Secure Power Calculation

**function** DGISPC(integer $t$, real $time$, real $\varepsilon$)
  **var**
    ▷ $X_{ij} = X_{ji}$ and $R_{ij} = R_{ji}$
    ▷ $X_{ij}$ and $R_{ij}$ are known line characteristics
    $X_{t-3,t-2}, \ldots, X_{t+2,t+3}$ : real;
    $R_{t-3,t-2}, \ldots, R_{t+2,t+3}$ : real;
    $I_{t-2}, \ldots I_{t+2}$ : boolean;
    $\hat{P}_{t-2}, \ldots, \hat{P}_{t+2}$ : real;
    $V_{t-3}, \ldots, V_{t+3}$ : real;
    $\theta_{t-3}, \ldots, \theta_{t+3}$ : real;
    $X, R$ : real :=0.0,0.0;
    $P_{cal}$ : real;
    $i, j, k$ : integer;
    $term1, term2$ :real :=0.0,0.0; ▷ Only for clarity
  **begin**
    get values $\{\hat{P}_{t-2}, \ldots, \hat{P}_{t+2}\}$ for given $time$;
    get values $\{\theta_{t-3}, \ldots, \theta_{t+3}\}$ for given $time$;
    get values $\{X_{t-3}, \ldots, X_{t+3}\}$ for given lines;
    **for** $i := t - 2$ **to** $t + 2$ **do**
      $X := X_{i-1,i}$;
      $R := R_{i-1,i}$;
      $term1 := R\{V_{i-1} - V_i \cos(\theta_{i-1} - \theta_i)\}$
      $term2 := XV_i \sin(\theta_{i-1} - \theta_i)$
      $P_{i-1,i} := \frac{V_{i-1}}{X^2 + R^2}[term1 + term2]$;
      $term1 := R\{V_{i-1} \cos(\theta_{i-1} - \theta_i) - V_i\}$
      $term2 := XV_{i-1} \sin(\theta_{i-1} - \theta_i)$
      $P_{i,i+1} := \frac{V_i}{X^2 + R^2}[term1 + term2]$;
      **if** $P_{i-1} + P_i - P_{i,i_1} < \varepsilon$ **then**
        $I_i := true$;
      **else**
        $I_i := false$;
      **end if**
    **end for**
    **if** $(\sim I_{t-1} \wedge \sim I_{t+1}) \vee (\sim I_t \wedge (\forall k \neq t)(I_k))$ **then**
      $P_{t-1,t} := P_{t-2,t-1} + \hat{P}_{t-1}$;
      $P_{t,t+1} := P_{t+1,t+2} + \hat{P}_{t+1}$;
      $P_{cal} := P_{t,t+1} - P_{t-1,t}$;
    **else**
      $P_{cal} := \hat{P}_t$;
    **end if**
    **return** $P_{cal}$;
  **end**;
**end function**

# 8. REMARKS

While much work has been reported in the literature on the proposed smart grid, there is still much work to be done on grid security. Because the grid is a CPS, security is not simply a matter of cyber security or physical security. The intertwining of the two leads to a much more complex security problem. A malicious house on a common distribution line could mount a fake power injection attack that could be nondeducible from the cyber messaging and from physical measurements. Electrical power cannot easily be "signed" as to its source. This work shows that in a small distribution network with fewer than seven nodes, it is entirely feasible for a malicious node to launch a fake power injection attack that would easily be detected by physical measurements, but the source could not be identified.

In the proposed smart grid as envisioned by the FREEDM Project, it is possible for a malicious node to launch a fake power injection attack in such a way that the source of the attack cannot be determined by purely cyber monitoring or purely physical monitoring. An intelligent attacker could easily hide behind the privacy requirements of the system and the inherent nondeducibility so introduced. Indeed, if the DGI trusts all the nodes, the attack will be undetected and untraced; however, if there is doubt in the veracity of the messages and reports of the readings from the nodes, an outside verification method, the proposed verifier, could determine if one node in seven is malicious. The verifier would have enough information to report back this fact and the identity of the attacker without violating the privacy constraints of the system.

However, the smart grid is a CPS and the fact that the physical part of the system can be observed can be used to break the nondeducibility of the attack by using the cyber messages in combination with physical attestations to create a situation where the attack would disrupt the physical invariants in a unique pattern if a verifier has access to the measurements reported by seven nodes. If the attacker is naive or if the attacker is clever, the verifier can still use attestation to form invariants and determine the source of the attack.

Indeed, the verifier can use the same information to calculate the correct value for the power generated by the attacker without violating the privacy of any house. Using this method, a network of at least seven houses is safe from the single center node attempting a fake power injection attack. If more nodes form the electrical smart grid grouping, the set of seven concurrent nodes can be relabeled to allow all but the three nodes on either the input or output side of the group to be individually verified. This technique shows promise for the possible extension to other network topologies. However, such an extension is outside the scope of this work.

# 9. DISCUSSION

A few simple examples of MSDND(ES) will help to more clearly explain how it can be applied and what sorts of information it can reveal. While designed for complex systems, MSDND(ES) can also be applied to simple situations. In order to look at MSDND(ES) and the ramifications, a few small "thought experiments", or "gedankenversuch", might be helpful.

## 9.1. GEDANKEN OR THOUGHT EXPERIMENT

In Section 3.3.4 a simple polynomial time reduction is presented to model any Sutherland ND(ES) using Multiple Security Domains Nondeducibility. The situation is not symmetric. MSDND(ES) can model any ND(ES) situation, *but the reverse is not the case.* There are problems that are MSDND(ES) secure where ND(ES) is indeterminate. One such case is the Gedanken experiment, "The Two Coin Dilemma" presented here.

**9.1.1. Sutherland ND(ES).** To demonstrate Sutherland ND(ES), it is helpful to perform a "Thought Experiment" that could easily be done as a real world demonstration. Imagine two security experts, Hal and Lou, waiting for a conference to begin. To pass the time, the two decide to explore nondeducibility by matching quarters. They agree that if they flip matching results ($\varphi$), heads-heads or tails-tails, Hal gets both quarters. If not, Lou gets both.

To make it more interesting, the two decide to demonstrate to everyone nondeducibility as they flip quarters. Hal flips his coin in the usual way and hides the result from Lou. Lou flips his coin on the table for all to see.

Let:

| | |
|---|---|
| $Q_H(w)$ | Hal's quarter (either "heads" or "tails") |
| $Q_L(w)$ | Lou's quarter |
| $\varphi =$ | $(Q_H(w) ='' heads'' \wedge Q_L(w) ='' heads'') \vee (Q_H(w) ='' tail'' \wedge Q_L(w) ='' tails'')$ |
| $w_m \in W$ | The Kripke frame worlds on which the quarters match |
| $w_n \in W$ | The Kripke frame worlds on which the quarters do not match |
| $V_H(w)$ | Hal's valuation function of which world (essentially the valuation of $\varphi$) |
| $V_L(w)$ | Lou's valuation function of which world (essentially the valuation of $\varphi$). |
| $H, T$ | Hal's coin "heads" or "tails" |
| $h, t$ | Lou's coin "heads' or "tails" |
| $\|X$ | trace restrictor, restricts the trace to security domain $X$ |
| $Tr$ | set of all valid traces |
| $\tau_X$ | System trace seen by $X$ |
| $\tau \in Tr$ | a particular valid trace |

For this experiment, ND(ES) holds for Lou if:

$$(\forall z \in \{heads, tails\}, \exists w_m \in W : V_H^{-1}(z) = w_m) \wedge [\exists w_n \in W : (V_H(w_n) = z) \wedge (V_L(w_m) = V_L(w_n))]. \tag{9.1}$$

**Theorem 27.** *Who wins, the quarters match or do not match, is not ND(ES) from Hal.*

*Modal Proof.*

Because Hal can see his quarter, his evaluation of match/mismatch depends solely upon the value of Lou's quarter. Without loss of generality, assume Hal has flipped "heads". Hal therefore knows that if Lou has flipped "heads", $w = w_M$ and $w = w_N$ otherwise. From his viewpoint, Hal can deduce the outcome.

**Case 27.1.** *Lou flips "heads".*

1. $Q_H(w)$ = heads  Hal has flipped heads
2. $Q_L(w)$ = heads  Lou has flipped heads
3. $w = w_m \rightarrow \varphi$  The coins match and Hal deduces he wins.

**Case 27.2.** *Lou flips "tails".*

1. $Q_H(w)$ = heads  Hal has flipped heads
2. $Q_L(w)$ = tails  Lou has flipped tails
3. $w = w_n \rightarrow \sim\varphi$  The coins do not match and Hal deduces he loses.

In either case, Hal is able to correctly deduce the outcome. Therefore; who wins, the quarters match or do not match, is not ND(ES) from Hal. □

*Trace Based Proof.*

The trace based proof follows the same reasoning. □

**Theorem 28.** *The state of $\varphi$, or the match/mismatch of the coins, is ND(ES) from Lou.*

*Proof.*

Without loss of generality, assume Hal has flipped heads. The proof can be divided into two cases depending upon what Lou has flipped.

**Case 28.1.** *Lou flips "heads".*

1. $Q_H$ = heads  Hal has flipped heads
2. $Q_L$ = heads  Lou has flipped heads
3. $w = w_m \rightarrow \varphi$  The coins match and Hal deduces he wins
4. $V_L(w_m) = Q_L$ = heads  Lou's coin is "heads" if they match
5. $V_L(w_n) = Q_L$ = heads  Lou's coin is "heads" if they don't match
6. $V_L(w_m) = V_L(w_n)$  Lou sees the same thing in either case
7. $\varphi$ is ND(ES)  $\varphi$ is nondeducible for Lou.

**Case 28.2.** *Lou flips "tails".*

| | | |
|---|---|---|
| 1. | $Q_H$ = heads | Hal has flipped heads |
| 2. | $Q_L$ = tails | Lou has flipped tails |
| 3. | $w = w_n \rightarrow \sim\varphi$ | The coins do not match and Hal deduces he loses |
| 4. | $V_L(w_m) = Q_L$ = tails | Lou's coin is "tails" if they match |
| 5. | $V_L(w_n) = Q_L$ = tails | Lou's coin is "tails" if they don't match |
| 6. | $V_L(w_m) = V_L(w_n)$ | Lou sees the same thing in either case |
| 7. | $\varphi$ is ND(ES) | $\varphi$ is nondeducible for Lou. |

In either case, Lou is unable to correctly deduce the outcome. Therefore, ND(ES) holds for Lou. □

*Alternate Trace Based Proof.*

Hal sees he has flipped "heads" but no one else has seen Hal's coin. After Lou flips his coin for all to see, there are two valid possibilities which are known to Hal.

**Case 28.3.** *Lou flips "heads".*

The set of valid traces for Lou is reduced to $Tr = \{Hh, Th\}$. When the trace restrictor is applied to each trace the results are $[(\{Hh\}|L = h) \wedge (\{Th\}|L = h)] \rightarrow \tau_L = \{h\}$. Regardless of what Hal has flipped, Lou sees only $\{h\}$.

**Case 28.4.** *Lou flips "tails".*

The set of valid traces Lou can see is reduced to $Tr = \{Ht, Tt\}$. When the trace restrictor is applied to each trace the results are $[(\{Ht\}|L = t) \wedge (\{Tt\}|L = t)] \rightarrow \tau_L = \{t\}$. Regardless of what Hal has flipped, Lou sees only $\{t\}$.

Therefore, in either case Lou has no information about what Hal has rolled and the theorem, 'The state match/mismatch is ND(ES) from Lou." holds. □

Suppose the game is changed to have Hal flip his coin on the table while Lou hides his result. The nondeducibility, ND(ES) of $\varphi$, is simply reversed.

**Theorem 29.** *The results are the same if Lou flips his coin the regular way and Hal flips his onto the table.*

*Proof.* The reasoning is an exact mirror of Theorem 27 and Theorem 28 and is not given here. □

An interesting situation occurs if both look at their coins, but do not announce the results. In this case, ND(ES) is symmetric and holds for both Hal and Lou at the same time. This result was hinted at by Sutherland and McLean [8] [7].

**Theorem 30.** *Sutherland's ND(ES) is symmetric if both hide their coins after looking at them.*

*Proof.*

1. ND(ES) holds for Lou by Theorem 28
2. ND(ES) holds for Hal by Theorem 29
3. ND(ES) is symmetric

Therefore, Sutherland's ND(ES) is symmetric if both hide their coins after looking at them. □

**9.1.2. The Two Coin Dilemma.** But what if Hal and Lou agree to both flip their coins and pause without looking at their coins? What is the state of ND(ES) *before* Lou or Hal know their results? This changes the situation dramatically. Does ND(ES) hold for both Hal and Lou or neither of them?

**Theorem 31.** *Sutherland ND(ES) cannot be evaluated until either Hal or Lou looks at their coin.*

It has already been shown that once Hal or Lou sees their own coin, ND(ES) holds for that person. The dilemma collapses into simple ND(ES) at that point.

*Proof.*

Without loss of generality, for Hal the first clause of Sutherland ND(ES) is: $(\forall z \in \{heads, tails\}, \forall w \in W : V_H^{-1}(z) = w)$. However, Hal is unable to perform the evaluation $V_H(w)$ to determine the outcome of this clause. ND(ES) neither holds nor fails, it simply cannot be applied. From Lou's viewpoint, the same reasoning holds and there is a symmetric failure of Sutherland's ND(ES). □

*Trace Based Proof.* Recall from Section 2.3, equation 2.4 the definition of trace based ND(ES) is: $ND(ES) = \forall \tau_L, \tau_H \in Tr : \exists \tau \in Tr : \tau|L = \tau_L|L \wedge \tau|HI = \tau_H|H$ (equation 2.4). No one has seen either coin, so from either Hal's or Lou's security domain, the trace is empty because no input or output actions have occurred.

This is an interesting situation. Looking at equation 2.4 one clause at a time must indicate the status of ND(ES). $\tau_H, \tau_L,$ and $\tau$ are all empty and *by definition* elements of any set, specifically $Tr$. Any restrictor applied to an empty trace will return a result of empty. It would appear that ND(ES) is *vacuously true.* It is logically correct to infer anything at all. Apparently ND(ES) has broken down and cannot be evaluated until the trace is populated with something.

□

Once Hal knows either "heads" or "tails", Sutherland ND(ES) holds for Lou and not Hal as before. But ND(ES) *cannot* even be evaluated before either knows their results. Intuitively, the result must be nondeducible, but how can ND(ES) be constructed to reflect the situation? ND(ES) relies upon the ability to perform the implied evaluation of both events by something in the model, even if that is only some phenomenon. Lacking any evaluation, ND(ES) breaks down.

**9.1.3. Multiple Security Domains Nondeducibility MSDND(ES).** Now, suppose Hal and Lou use MSDND(ES) to analyze the same game. Notice, by the definition

of a logical statement the condition $\varphi$ **xor** $\sim\varphi$ must hold. We can simplify all proofs of MSDND(ES) for $\varphi$ to showing the final clause, $\nexists\ V_{\varphi}(w)$. For simplicity, the same nomenclature will be used as before. In the first game where Hal flips his coin and looks at it before he announces the result but Lou flips his on the table for all to see, MSDND(ES) holds for Lou if:

$$(\forall w \in W)(\varphi \textbf{ xor } \sim\varphi) \wedge [w \vDash \Box(\nexists\ V_L(\varphi))]. \tag{9.2}$$

Notice: by the definition of a logical statement, the condition $\varphi$ **xor** $\sim\varphi$ must hold. This means MSDND(ES) can be simplified to showing the final clause,$\nexists\ V_{\varphi}(w)$ is true.

As before, the situation for Hal mirrors that of Lou. After the first flip:

**Theorem 32.** *The state match/mismatch is not MSDND(ES) from Hal.*

*Proof.* Because Hal can see his quarter, his evaluation of match/mismatch depends solely upon the value of Lou's quarter. Without loss of generality, assume Hal has flipped "heads". Hal therefore knows that if Lou has flipped "heads", $w = w_M$. From his viewpoint, Hal can deduce the outcome.

**Case 32.1.** *Lou flips "heads".*

1. $Q_H(w)$ = heads  Hal has flipped heads
2. $Q_L(w)$ = heads  Lou has flipped heads
3. $w = w_m \rightarrow \varphi$  The coins match and Hal deduces he wins.

**Case 32.2.** *Lou flips "tails".*

1. $Q_H(w)$ = heads  Hal has flipped heads
2. $Q_L(w)$ = tails  Lou has flipped tails
3. $w = w_n \rightarrow \sim\varphi$  The coins do not match and Hal deduces he loses.

In either case, Hal is able to correctly deduce the outcome. Therefore, MSDND(ES) does not hold for Hal because Hal possesses a valuation function for $\varphi \rightarrow \exists V_H(\varphi)$. □

**Theorem 33.** *The state match/mismatch is MSDND(ES) from Lou.*

*Proof.*

**Case 33.1.** *Lou flips "heads".*

| | | |
|---|---|---|
| 1. | $Q_H$ = heads | Hal has flipped heads |
| 2. | $Q_L$ = heads | Lou has flipped heads |
| 3. | $w = w_m \rightarrow \varphi$ | The coins match and Hal deduces he wins |
| 4. | $V_L(w_m) = Q_L$ = heads | Lou's coin is "heads" if they match |
| 5. | $V_L(w_n) = Q_L$ = heads | Lou's coin is "heads" if they don't match |
| 6. | $V_L(w_m) = V_L(w_n)$ | Lou sees the same thing in either case |
| 7. | $\nexists\, V_\varphi(w)$ | and $\varphi$ is nondeducible for Lou. |

**Case 33.2.** *Lou flips "tails".*

| | | |
|---|---|---|
| 1. | $Q_H$ = heads | Hal has flipped heads |
| 2. | $Q_L$ = tails | Lou has flipped tails |
| 3. | $w = w_n \rightarrow \sim\varphi$ | The coins do not match and Hal deduces he loses |
| 4. | $V_L(w_m) = Q_L$ = tails | Lou's coin is "tails" if they match |
| 5. | $V_L(w_n) = Q_L$ = tails | Lou's coin is "tails" if they don't match |
| 6. | $V_L(w_m) = V_L(w_n)$ | Lou sees the same thing in either case |
| 7. | $\nexists\, V_\varphi(w)$ | and $\varphi$ is nondeducible for Lou. |

In either case, Lou is unable to correctly deduce the outcome. Therefore, MSDND(ES) $\varphi$ holds for Lou. □

Apparently, Sutherland ND(ES) and MSDND(ES) produce the same result. This is to be expected because it has already been shown that Sutherland ND(ES) can be reduced in polynomial time to MSDND(ES). But what about the game where neither Hal or Lou look at their coin? At that point, MSDND(ES) holds for both Hal and Lou.

**Theorem 34.** *Multiple Security Domains Nondeducibility holds for Hal before either looks at their coin.*

*Proof.*

Without lose of generality, assume Hal flips "heads". At this point, Hal *does not know he flipped "heads".*

**Case 34.1.** *Lou flips "heads" but does not know it.*

| | | |
|---|---|---|
| 1. | $Q_H ='' heads''$ | Hal has flipped "heads" but does not know that. |
| 2. | $Q_L ='' heads''$ | Lou has flipped "heads" but does not know that. |
| 3. | $\nvdash Q_H \rightarrow \nvdash V_H(w) \rightarrow \nvdash V_H(\varphi)$ | Hal cannot know who won. |

**Case 34.2.** *Lou flips "tails" but does not know it.*

| | | |
|---|---|---|
| 1. | $Q_H ='' heads''$ | Hal has flipped "heads" but does not know that. |
| 2. | $Q_L ='' heads''$ | Lou has flipped "tails" but does not know that. |
| 3. | $\nvdash Q_H \rightarrow \nvdash V_H(w) \rightarrow \nvdash V_H(\varphi)$ | Hal cannot know who won. |

Hal does not have a valuation function for $\varphi$, $\nvdash V_H(\varphi)$, and therefore MSDND(ES) for $\varphi$ holds for Hal. Mirror reasoning leads to the same conclusion for Lou. Again, a symmetric result as is expected. □

**Theorem 35.** *Multiple Security Domains Nondeducibility holds for Lou before either looks at their coin.*

*Proof.*

The proof mirrors the same logic as Theorem 34 □

**9.1.4. Schrödinger's Cat and ND(ES).** In discussing this famous Gedanken experiment, mathematics and physics will be kept to a minimum. This will lead to some bending of the exact nature of the experiment. However, this can be tolerated because the intent is to look at information flow, not wave mechanics. The background will be kept as brief as possible.

In the experiment, see Figure 9.1, a box is constructed with a deadly vial of cyanide suspended above a vat of acid. If a specific atom of uranium undergoes spontaneous fission, the vial is dropped into the acid and the generated gas kills the cat. This fission event is

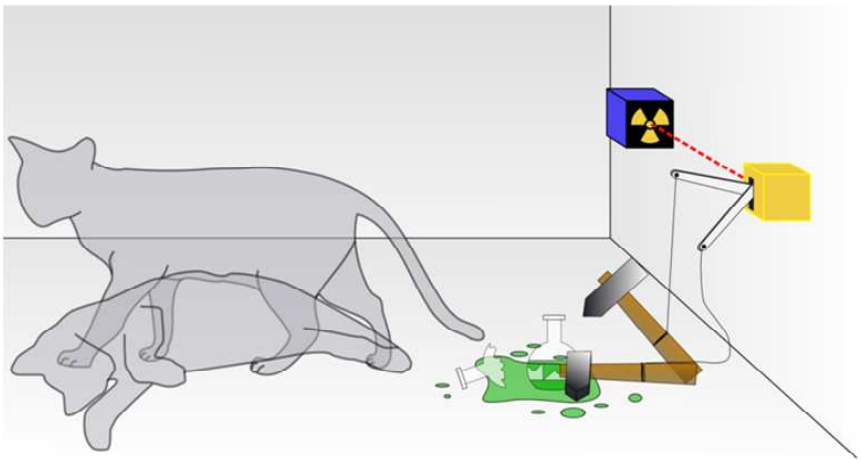


Figure 9.1. Schrödinger's Cat

unpredictable and totally random. The cat is placed in the box and the box is sealed in such a way to cut off its contents from any and all interactions with the universe. Before the box is sealed, the contents have a wave function, $\Psi_{cat}$, that is part of the wave function of the universe, $\Psi_U$. When the box is sealed, the two wave functions are uncoupled and cannot interact with each other. (This is key for quantum mechanical reasons that are unimportant to this discussion.) Time passes both inside and outside of the box.

Dr. Schrödinger posed the famous question: is the cat alive or dead? For purposes of discussion, let $\varphi$ be "The cat breathed five minutes after the box was closed."; that is, the cat was alive and did something. Because the box is sealed off *completely* from the universe, the wave function inside the box is decoupled from the universe, the cat is neither alive nor dead but somehow a combination of the two until the box is opened (and the two wave functions couple again, technically the cat's state collapses to either alive or dead) or in any way interacts with the universe.

This presents two completely partitioned security domains, Dr. Schrödinger (and the universe) $SD^S$ and the cat (and everything inside the box) $SD^C$. For the sake of discussion, assume many different things happen while the box is closed. In $SD^S$ inputs

and outputs happen until the box is finally opened ($BoxOpen$). The states can be labeled $s_1, s_2, \ldots, s_n, \ldots, BoxOpen$, although the detailed states are not of interest until state $BoxOpen$ where the box is opened. In $SD^C$ the events are problematic. Because of the nature of the experiment, nothing can be said about the events inside the box. The quantum states are not collapsed and therefore any statements involving the events are indeterminate at best until $\ldots, BoxOpen$ at which point the previous events inside the box have meaning.

**Theorem 36.** *Before the box is closed, events inside the box are not ND(ES) secure from the universe.*

*Proof.* Obviously, Dr. Schrödinger can walk over and look into the box. On a more technical level, $\Psi_{cat}$ is coupled with $\Psi_U$ and therefore observable by some means. □

**Theorem 37.** *While the box is closed off from the universe, it is not possible to determine if the events inside are ND(ES) secure from the universe.*

*Proof.* Suppose not. Suppose the theorem is false, then it is reasonable to decide the cat is either alive ($\varphi$) or dead ($\sim\varphi$)[12].

Let:

| | | |
|---|---|---|
| $\varphi$ | be | "The cat breathed five minutes ago." |
| $\sim\varphi$ | be | "The cat has not breathed for five minutes." |
| $SD^S$ | be | the security domain of the outside universe and Dr. Schrödinger. |
| $SD^C$ | be | the security domain of the box and the cat. |
| $\mathbb{V}^S_\varphi(w)$ | be | the Dr. Schrödinger's valuation of $\varphi$. |
| $\mathbb{V}^C_\varphi(w)$ | be | the cat's valuation of $\varphi$. |

To show the state of the cat is ND(ES) we must show:
$\forall z = \left(\mathbb{V}^C_\varphi(w)\right)^{-1} : \mathbb{V}^C_\varphi(w') = z \wedge \left(\mathbb{V}^S_\varphi(w) = \mathbb{V}^S_\varphi(w')\right)$. But for Dr. Schrödinger to evaluate the state of $\varphi$, he must somehow see inside the box. This violates the terms of the experiment. For the cat to let it be known that it is breathing, it must communicate with the outside world which also violates the experiment. This is a contradiction, therefore the

[12] This is where the discussion does not exactly follow quantum mechanics.

theorem is true and it is not possible to determine the ND(ES) state of the events inside the box. □

**Remarks 2.** Most situations where ND(ES) is evaluated require that one partition or the other have a "god's" view of both partitions. This allows information to flow from the *LOW* partition to the *HIGH* partition.

**Theorem 38.** *While the box is closed off from the universe, it* is *possible to determine if the events inside are MSDND(ES) secure from the universe.*

*Proof.*

Let:

| | | |
|---|---|---|
| $\varphi$ | be | "The cat breathed five minutes ago." |
| ~$\varphi$ | be | "The cat has not breathed for five minutes." |
| $SD^S$ | be | the security domain of the outside universe and Dr. Schrödinger. |
| $SD^C$ | be | the security domain of the box and the cat. |
| $\mathbb{V}^S_\varphi(w)$ | be | the Dr. Schrödinger's valuation of $\varphi$. |
| $\mathbb{V}^C_\varphi(w)$ | be | the cat's valuation of $\varphi$. |

1. $\varphi$ **xor** ~$\varphi$ def. of wff
2. ~$\exists\mathbb{V}^S_\varphi(w)$ terms of the experiment

By sets 1 and 2:

$$\forall w\ inW : w \vdash (\varphi \textbf{ xor } \sim\varphi) \wedge \big(w \vDash \big(\sim\exists\mathbb{V}^S_\varphi(w)\big)\big).$$

The state of the cat cannot be determined by Dr. Schrödinger without violating the experiment, but the state is MSDND(ES) secure.

□

**Remarks 3.** Most situations where ND(ES) is evaluated require one partition or the other have a "god's" view of both partitions. This allows information to flow from the *LOW* partition to the *HIGH* partition.

**9.1.5. Results of Gedankenversuch.** These little thought experiments, "gedanken-versuch", show that Multiple Security Domains Nondeducibility produces correct results when Sutherland's ND(ES) cannot even provide any results. This is a critical result for this dissertation. This dissertation has already shown that MSDND(ES) can easily model any system where Sutherland's nondeducibility holds. If it had turned out that the reverse is true, that ND(ES) could model any system that can be modeled by MSDND(ES), then the only hope for the dissertation is to show that the new method is easier to implement. Instead, this dissertation shows that MSDND(ES) can indeed model systems where traditional nondeducibility cannot be evaluated.

This dissertation has a very simple idea at its heart: cyber security methods do not work well for CPS nor do physical security methods. What is needed is an entirely different point of view. To secure a CPS the cyber system must be secured, the physical system must be secured, information flows must be secured and controlled, and lastly the knowledge leaked to an outside agent who is able to simply observe the CPS must also be understood and secured. Even so, there is no guarantee that nothing has been missed.

The more traditional security methods are useful to understand the actors, subjects and objects, as well as any inherent security domains of the system. These traditional methods are not enough to insure the security of a CPS, but many times they can address the needs of the cyber side of the system. A quick modeling of the system usually leads to false starts and retracing of steps already done, but the effort can lead to critical insights into the nature of the CPS.

## 9.2. LABELED TRANSITION SYSTEMS

It is tempting to create a security model for a CPS over a Labeled Transition System (LTS); however, this is problematic when dealing with even simple real world systems. To correctly build the frame $\mathfrak{F}$ *all* the possible transitions must be defined. Even denoting all the possible states to build a single world is difficult in the real world, but trying to correctly determine all the possible changes and the corresponding transitions is virtually impossible.

There are only two reasonable ways to deal with the complexity of the transitions, either simplify the model until very few transitions are possible or allow all transitions to occur.

If the transitions are known and an LTS can be built, the modal operators take on a different form and become directly related to the labeled transitions. Briefly, $\Box\varphi$ becomes $[i]\,\varphi$ where $i$ is the number of transitions in the statement. The $\diamond\varphi$ operator becomes $\langle i\rangle\,\varphi$ in the same manner. The behavior of an LTS is drastically different from a complete transition frame. Because it is rarely useful to denote all the allowed states and transitions for a CPS, this dissertation, and indeed MSDND itself, is based solely upon complete transition Kripke frames thus removing the requirement to build complex transition tables or diagrams.

## 9.3. TIME AND TRACES IN KRIPKE FRAMES

While it does not have a strict bearing upon this dissertation, the concepts of time and system traces of a model built over Kripke frames is interesting. There are two obvious ways to deal with time that can produce an acceptable system trace.

**9.3.1. Time as a State Variable.** If time is treated as a state variable, the passage of time causes a transition from one world $w$ to another $w'$ on the frame. A corresponding change to a different state variable, for example "the brakes are applied", would lead to another transition where the time would not change. A model built upon this type of view would closely resemble a grand canonical ensemble of essentially static states joined by both time and state change transitions. A trace could be produced by following the transitions as one would expect.

**9.3.2. Time as Purely Transitional.** Another possibility would be to look at every "tick" of the system as forcing a transition. If the granularity of the "ticks" is small enough, such a rigid system could usefully model a CPS, but a method must be found to deal with time when the system is at rest. If each "tick" forces a transition, that is $w_1 R w_2, w_2 R w_3, \ldots$, then the simple solution would be to allow a time transition to return to the same world which would leave all state variable values unchanged. In this view, a trace could be manufactured by following the transitions in a strict time order as one would

expect. Obviously, the choice of methods to deal with time must be independent of the actual CPS and the choice would be made based solely upon which view of time is most useful.

## 9.4. THE ADVANTAGES OF MSDND(ES)

The main advantages of MSDND(ES) fall into two main categories, usefulness and semantics. This method can describe models that cannot be easily described with Sutherland's ND and handle constraints that could otherwise interfere with the usefulness of the model. On the other hand, MSDND(ES) allows the model to examine difficult questions with less semantic distance than many other methods and with quite a bit of flexibility. Another intriguing possibility of MSDND(ES) is the possibility of localizing the actual source of information flow by analyzing a trace to determine where a breach has occurred. Unfortunately, this will have to wait for future work.

## 9.5. EXTENDED NONDEDUCIBILITY

Multiple Security Domains Nondeducibility can be trace based, but it was not designed that way. Traced based security can require the model to be run multiple times in order to examine the trace under different conditions. True, this could be done by trace restrictors such as used in NI or NF, but this is not required. Because MSDND is based upon a Kripke frame, there is no need to wait for the completion of a set of system actions to examine a trace. It is possible with some models to examine the MSDND(ES) status of the system as it evolves. This allows for the possibility of future systems that automatically perform actions to help hide internal actions [58] in order to minimize the information flow when the physical side of the CPS is observed.

Some systems forbid access to sensitive information to preserve privacy such as the proposed electrical smart grid [19]. A homeowner may very well want to preserve his or her privacy by refusing to allow his neighbors to read the meter attached to his or her house to determine the precise electrical generation and/or demand. This makes traditional

information flow security difficult, or impossible, to evaluate, see Sections 7 and 9.1. Sutherland ND(ES) cannot be evaluated in these situations but MSDND(ES) can.

## 9.6. STRONG AND WEAK NONDEDUCIBILITY

Another view of the relationship between Sutherland's Nondeducibility and MSDND(ES) is to look at the constraints upon the systems each can model. In order to evaluate ND(ES), either the system must be constrained to a limited number of well defined traces or to a specific type of Kripke frame where all values on all worlds can be evaluated. This is a relatively strong model and works very well in describing most CPS. But CPS are sometimes not very well defined or have components that are not completely understood at the time the model is constructed. This leads to the distinct possibility that some transitions may not be correctly modeled leading to traces that were not expected or to the possibility that a required logical expression might not have an associated valuation in the model. In these cases, ND(ES) will fail to properly model the actual system as was seen in the example of the drive-by-wire automobile, see Section 5.

MSDND(ES) is a weaker model than ND(ES) with fewer constraints. Because MSDND(ES) has fewer constraints, it is more useful in situations where the system is not well-behaved or not as well understood. For example, MSDND(ES) is shown to produce a model that behaves more like the CPS of the drive-by-wire automobile. MSDND(ES) is better able to model systems where the constraints of the system do not allow for some of the valuation functions required such as the electrical smart grid as presented in Section 7, the coin flip game presented in Section 9.1.2, or Schrödinger's Cat presented in Section 9.1.4. In all of these cases, the constraints of the actual CPS do not allow some of the desired valuation functions and therefore present serious problems for Sutherland's ND(ES). A weaker model with fewer constraints, MSDND(ES), is required.

This relationship of a weaker to a stronger model also explains why it is possible to reduce Sutherland's ND(ES) to a weaker MSDND(ES) because this is a relaxation of constraints. This should not be taken as a criticism of ND(ES) but more as an extension of

Nondeducibility from the purely cyber field of system security to a broader usefulness in the study of CPS.

## 9.7. MSDND(ES) AND SEMANTIC DISTANCE

Early efforts to describe CPS required the system to be defined in rigid terms. HRU requires the system to be deconstructed into *subjects*, *objects*, and *rights*. The actions performed upon these entities were limited and in some cases irreversible. While this produces useful results in some limited cases, the model does not clearly reflect the system under examination. The semantic distance is large. The rigid model produced does not correspond well with the conceptual view of the system. The BLP, Lipner, and Biba models with their added complexity bring the model more in line with the conceptual system, but still leave much to be desired when applied to CPS . These systems were never meant to deal with information flows and physically observable systems.

Information flow security models do much towards closing the semantic distance between the model and the conceptual system. These systems relax the rigid requirements of the earlier models while allowing one to examine the more subtle ways in which information flows can disrupt security. In short, the question set that can be answered is much richer.

Nondeducibility and Multiple Security Domains Nondeducibility close the semantic gap even more than other information flow security models. These models can formulate, and answer, a question set that is richer and much closer to both the conceptual CPS and the actual CPS . Because the valuations are more flexible, virtually any question that can be framed as a logical expression can be dealt with.

## 9.8. MSDND(ES), ND(ES), AND LEVELS OF ABSTRACTION

When dealing with conceptual models of actual CPS issues of semantic abstraction levels become increasingly important. S. I. Hayakawa introduced the concept of the ladder of abstraction [60], see chapter 10, to deal with different levels of semantic abstraction in a clearer manner than Alfred Korzybski's structural differential [61] [62]. The historical

security models in Section 2 are constrained to deal with questions on a single abstraction level, typically the level closest to the model. This is too simplistic for information flows, especially when considering interactions between the CPS and society. Both ND(ES) and MSDND(ES) are designed to deal with valuations on Kripke frames regardless of the level of semantic abstraction. This allows Nondeducibility techniques expressed over Kripke frames to deal with questions of information flow between different levels of semantic abstraction by framing the questions involved as simple or compound logical expression. This is particularly useful when ND(ES) and MSDND(ES) are extended into other disciplines and not limited to CPS. Traditional cyber security methods cannot be extended in this manner.

# 10. CONCLUSIONS

## 10.1. TRADITIONAL SECURITY METHODS

As stated numerous times in this dissertation, the traditional security methods such as HRU, BLP, Lipner, NF, and NI are not sufficient to secure most CPS. However, these tools are vital for developing the cyber security necessary for total security of CPS and are key to creating a structure capable of resisting attacks. Working through the process of correctly using these tools can be tedious and prone to redundant efforts; however, persistent effort on this phase is key to building a useful model.

Information flow security is critical in CPS. Early attempts to deal with the issues of IFS such as Noninterference and Noninference provide a way to model simple attacks with effects that are obvious. However, attackers are becoming more sophisticted and so are the methods to model more subtle information flows. Sutherland introduced the idea of Nondeducibility as a model built over a Kripke frame using modal logic methods. This dissertation discusses a new method of nondeducibility, MSDND(ES), and presents a polynomial time method to reduce any model that is ND(ES) to one that is MSDND(ES). This method is used to examine models of a number of different MSDND(ES).

## 10.2. PHYSICAL SECURITY CLUES

All CPS can be observed. This apparent weakness can be shown to be a very useful tool when properly understood. If the cyber security is viewed with the correct level of “trust but verify”, physical observations can be paired with cyber monitoring to provide physical attestations, but there must be the understanding that cyber security alone is not the answer to every security problem. The operator (human) or the security monitoring system *must*

be willing to use the physical system to verify the cyber system. This key point of social engineering was one of the key reasons the Stuxnet attack was successful.

## 10.3. BIT/BUT MODAL LOGIC AND CPS

The BIT logic introduced by Liau [9] proves very useful in discussing the role of trust, either human or computer agent, in CPS. In much of the literature, when trust or belief is discussed, it can take pages of text to explain relatively simple concepts. With BIT or BUT logic, these complex trust/belief relationships can be treated mathematically which not only reduces the amount of text used, but can translate imprecise language into a symbolic language amenable to mathematical proofs. This clarity facilitates the discovery of new information.

## 10.4. DRIVE-BY-WIRE AUTOMOBILE

Like most modern automobiles, the Toyota Prius as a drive-by-wire system can present some interesting security issues. The system fits the multi-level BLP model, but the requirements for a more secure subject to lower its security level to make the system function implies a large amount of trust in the entity. In the Lipner model the $Corporation$ subject is not trusted, resulting in the system operations being inconsistent with known operations. Both Noninference and Nondeducibility are information flow models that describe the ability of the driver to ascertain how the vehicle is being operated. Specifically, the system is Nondeducibility secure with respect to the driver which means the driver cannot ascertain if $Corporation$ remote operations or the $Car$ is controlling the behavior.

If the owner subscribes to OnStar, Toyota Safety Connect, or some similar service, the driver must trust the service. We have shown that in hazardous situations or in remote operations the driver is **not** in control of the automobile. There is nothing the driver can do in these situations but trust that all is well. Such concerns spread beyond the systems studied here; the recent Stuxnet worm [15] had a similar effect of blinding the system operator from

the actual CPS operation. As such, this dissertation's analysis is indicative of the type of analysis needed before large-scale adoption of cyber-physical services.

This section models a particular CPS, the drive-by-wire automobile using an number of security models from BLP to the IFS models such as Noninterference, Noninference, and trace-based Nondeducibility. If a model is used that does not take into account IFS, the results do not clearly describe the observed actions of the CPS. The IFS models more closely reflect the reality of the drive-by-wire automobile, but there are still issues.

The models as presented are all trace based. The CPS must be closely followed through the actions of interest and *then* the trace can be examined to insure that the particular IFS holds. For both NI and ND the model must be run *twice* under different security conditions and the resulting traces compared. Trace based information flow models do not offer the promise of real-time analysis of CPS.

As noted in Section 2.3, Sutherland Nondeducibility can best be expressed as a modal frame based model rather than a trace based model. With frame based ND models, the requirement to run the model multiple times can be relaxed. This would seem to eliminate all of the major issues with ND but in many cases it does not. Sutherland's ND requires the model to contain potential valuation functions for *all wff for all worlds.* This is often not the case with CPS. If valuation functions are missing, the model fails and ND cannot be determined one way or the other.

### 10.5. MSDND

The traditional view of security, the idea of "walling the bad guys out", is too simplistic. Viewing security domains as wholly contained within a threat space or within a less secure domain is inadequate as are the available tools. Restricting models to idealized partitions does not work well with cyber physical systems.

We have shown that multiple security domains, without the necessity of ideal partitions, is a more realistic model. We have shown that in CPS information leaks throughout the model by observation of the physical actions of the system. Our new

definition of MSDND(ES) can model traditional Nondeducibility as well as provide a definition of Nondeducibility that holds in CPS. Specifically, MSDND(ES) can easily model situations where critical information flow from one security domain to another is disrupted or denied altogether as in the Stuxnet worm attack.

We applied our model to a specific CPS, a drive-by-wire automobile, under real world conditions. Our model fits the CPS better than traditional Nondeducibility because it does not require us to partition the system into idealized domains that do not allow information flow between domains. Indeed, our model does not even need to address how the security domains interact once they have been properly defined. We have shown that we can relax the requirements of absolute domain partitioning and still model the system.

Furthermore, we have shown that since MSDND(ES) does not depend upon the ability to evaluate information flow between distinct and absolute partitions, our model does not require building complicated decision variables nor does it require access to the total input/output of the model. By relaxing the boundary conditions of the model, results are obtained by modal methods.

## 10.6. STUXNET

MSDND(ES) and BIT logic can be used to model Stuxnet type attacks. Such attacks rely on MSDND(ES) and the inherent trust placed in the components of CPS to hide critical information from electronic monitoring and from human operators. Others [14] have discussed how difficult it is to thwart specifically targeted attacks such as APTs. Because Stuxnet-like attacks do not make an effort to steal information, there is no need for the virus to connect with the INTERNET. Therefore, monitoring out-bound traffic does not help.

Because such an attack replays valid readings, any effort to find problems through internal inconsistencies is also doomed. It is not feasible to eliminate the human components in large scale operations of CPS; therefore APT attacks will often be successful via social engineering. Once such a virus is in place, detection is complicated by human trust in

electronic systems. If we expect the electronic monitoring to give us correct results, a low-level attack on the physical sensor-monitor communications such as Stuxnet will succeed.

The importance of Corollary 6.6.1 is clear. All CPS must also have physical monitoring that can be used to verify the operation of the electronic monitoring or the next Stuxnet type attack will also succeed. Verifying cyber security with low level physical monitoring can break the role of trust in MSDND attacks. In the case of Stuxnet, the simple addition of a physical read-out of the actual speed of the centrifuge would have broken the attack model if the human operator distrusted the cyber monitoring enough to verify the readings on the monitor.

Stuxnet type attacks can be broken. Consider the centrifuge system in light of Corollaries 6.6.1 and 6.6.1. If the centrifuge is equipped with a *physical* speedometer in addition to the cyber monitoring, the speedometer can be made to trip an audible alarm or a cyber alarm with physical diversity from the normal monitoring/control system. For example, the speedometer might close a hard-wired circuit to turn on a siren and flashing red light. This is equivalent to all entities having direct access to the valuation function $\mathbb{V}^0_\varphi(w)$. If this is true, then Theorem 6.6.1 holds and MSDND(ES) based attacks fail.

## 10.7. THE ELECTRICAL SMART GRID

While much work has been reported in the literature on the proposed smart grid, there is still much work to be done on grid security. Because the grid is a CPS, security is not simply a matter of cyber security or physical security. The intertwining of the two leads to a much more complex security problem. A malicious house on a common distribution line could mount a fake power injection attack that could be nondeducible from the cyber messaging and from physical measurements. Electrical power cannot easily be “signed” as to its source. This work shows that in a small distribution network with fewer than seven nodes, it is entirely feasible for a malicious node to launch a fake power injection attack that would easily be detected by physical measurements, but the source could not be identified.

In the proposed smart grid as envisioned by the FREEDM Project, it is possible for a malicious node to launch a fake power injection attack in such a way that the source of the attack cannot be determined by purely cyber monitoring or purely physical monitoring. An intelligent attacker could easily hide behind the privacy requirements of the system and the inherent nondeducibility so introduced. Indeed, if the DGI trusts all the nodes, the attack will be undetected and untraced; however, if there is doubt in the veracity of the messages and reports of the readings from the nodes, an outside verification method, the proposed verifier, could determine if one node in seven is malicious. The verifier would have enough information to report back this fact and the identity of the attacker without violating the privacy constraints of the system.

However, the smart grid is a CPS and the fact that the physical part of the system can be observed can be used to break the nondeducibility of the attack by using the cyber messages in combination with physical attestations to create a situation where the attack would disrupt the physical invariants in a unique pattern if a verifier has access to the measurements reported by seven nodes. If the attacker is naive or if the attacker is clever, the verifier can still use attestation to form invariants and determine the source of the attack. Indeed, the verifier can use the same information to calculate the correct value for the power generated by the attacker without violating the privacy of any house. Using this method, a network of at least seven houses is safe from the single center node attempting a fake power injection attack. If more nodes form the electrical smart grid grouping, the set of seven concurrent nodes can be relabeled to allow all but the three nodes on either the input or output side of the group to be individually verified. This technique shows promise for the possible extension to other network topologies. However, such an extension is outside the scope of this work.

# BIBLIOGRAPHY


[1] G. Howser and B. McMillin, "Modeling and reasoning about the security of drive-by-wire automobile systems," *International Journal of Critical Infrastructure Protection*, vol. 5, pp. 127–134, December 2012.

[2] ——, "A Multiple Security Domain Model of a Drive-by-Wire System," in *Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual*. Computer Software and Applications Conference, 2013, pp. 369–374.

[3] G. Howser, T. Roth, and B. McMillin, "Breaking Multiple Security Domains Nondeducibility on the Smart Grid," under consideration for IEEE Transactions on Dependable and Secure Computing (TDSC).

[4] R. Browne, "The Turing test and non-information flow," in *Research in Security and Privacy, 1991. Proceedings., 1991 IEEE Computer Society Symposium on*. IEEE, 1991, pp. 373–385.

[5] L. Reznik, "Which models should be applied to measure computer security and information assurance?" in *Fuzzy Systems, 2003. FUZZ'03. The 12th IEEE International Conference on*, vol. 2. IEEE, 2003, pp. 1243–1248.

[6] D. Clark, S. Hunt, and P. Malacaria, "Quantified interference: Information theory and information flow," *Workshop on Issues in the Theory of Security*, 2004.

[7] J. McLean, "Security models and information flow," in *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on*, may 1990, pp. 180 –187.

[8] D. Sutherland, "A model of information," in *Proceedings of the 9th National Computer Security Conference*. DTIC Document, 1986, pp. 175–183.

[9] C.-J. Liau, "Belief, information acquisition, and trust in multi-agent systems - A modal logic formulation," *Artificial Intelligence*, vol. 149, no. 1, pp. 31 – 60, 2003.

[10] L. Wu, J. Su, K. Su, X. Luo, and Z. Yang, "A concurrent dynamic logic of knowledge, belief and certainty for multi-agent systems," *Knowledge-Based Systems*, vol. 23, no. 2, pp. 162 – 168, 2010.

[11] S. A. Kripke, "A Completeness Theorem in Modal Logic," *The Journal of Symbolic Logic*, vol. 24, no. 1, pp. pp. 1–14, 1959.

[12] P. Blackburn, M. De Rijke, and Y. Venema, *Modal Logic*. Cambridge University Press, 2002, vol. 53.

[13] P. Ryan, "Mathematical Models of Computer Security," in *Foundations of Security Analysis and Design*, ser. Lecture Notes in Computer Science, R. Focardi and R. Gorrieri, Eds. Springer Berlin / Heidelberg, 2001, vol. 2171, pp. 1–62.

[14] P. Clawson, "Pragmatic Defense Against Today's Advanced Targeted Attacks," Seminar at Missouri University of Science and Technology, October 2013.

[15] N. Falliere, L. O. Murchu, and E. Chien, "W32.Stuxnet Dossier," http://goo.gl/dC8VT, 2010, [Online; accessed December 2011].

[16] G. Howser and B. McMillin, "A Modal Model of Stuxnet Attacks on Cyber-Physical Systems: A Matter of Trust," in *2014 8th IEEE International Conference on Software Security and Reliability (SERE)*. IEEE, 2014, to appear.

[17] G. Dhillon and J. Backhouse, "Current directions in IS security research: towards socio-organizational perspectives," *Information Systems Journal*, vol. 11, no. 2, p. 127, 2001.

[18] R. Akella and B. M. McMillin, "Modeling and verification of security properties for critical infrastructure protection," in *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*. ACM, 2013, p. 6.

[19] T. Roth and B. McMillin, "Physical Attestation of Cyber Processes in the Smart Grid," in *Critical Information Infrastructures Security*. Springer, 2013, pp. 96–107.

[20] M. Harrison, W. Ruzzo, and J. Ullman, "Protection in Operating Systems," *Communications of the ACM*, vol. 19, no. 8, pp. 461–471, 1976.

[21] J. Glasgow, G. Macewen, and P. Panangaden, "A logic for reasoning about security," *ACM Trans. Comput. Syst.*, vol. 10, no. 3, pp. 226–264, Aug. 1992.

[22] J. Jacob, "Basic Theorems about Security," *Journal of Computer Security*, vol. 1, pp. 385 – 411, January 1992.

[23] P. F. Syverson and J. W. Gray III, "The epistemic representation of information flow security in probabilistic systems," in *Computer Security Foundations Workshop, 1995. Proceedings., Eighth IEEE*. IEEE, 1995, pp. 152–166.

[24] S. B. Lipner, "Non-Discretionery Controls for Commercial Applications," *Security and Privacy, IEEE Symposium on*, vol. 0, p. 2, 1982.

[25] J. A. Goguen and J. Meseguer, "Security Policies and Security Models," *Proceedings of the 1982 IEEE Symposium on Security and Privacy*, pp. 11–20, 1982.

[26] A. Zakinthinos, "On The Composition Of Security Properties," Ph.D. dissertation, University of Toronto, 1996.

[27] D. von Oheimb, "Information Flow Control Revisited: Noninfluence = Noninterference + Nonleakage," in *Computer Security ESORICS 2004*, ser. Lecture Notes in Computer Science, P. Samarati, P. Ryan, D. Gollmann, and R. Molva, Eds. Springer Berlin / Heidelberg, 2004, vol. 3193, pp. 225–243.

[28] H. Mantel, "A uniform framework for the formal specification and verification of information flow security," Ph.D. dissertation, Universitätsbibliothek, 2003.

[29] M. Bishop, *Computer Security: Art and Science*. Addison-Wesley, 2003.

[30] T. Roth and B. McMillin, "Breaking Nondeducible Attacks on the Smart Grid," in *Seventh CRITIS Conference on Critical Information Infrastructures Security*. Seventh CRITIS Conference on Critical Information Infrastructures Security, 2012, (to appear).

[31] J. Madden, B. McMillin, and A. Sinha, "Environmental obfuscation of a cyber physical system-vehicle example," in *Computer Software and Applications Conference Workshops (COMPSACW), 2010 IEEE 34th Annual*. IEEE, 2010, pp. 176–181.

[32] J. Madden, "Security analysis of a cyber physical system : a car example," Master's thesis, Missouri University of Science and Technology, 2013. [Online]. Available: http://hdl.handle.net/10355/36259

[33] S. Popkorn, *First steps in Modal Logic*. Cambridge University Press, 1994.

[34] M. Gehrke, "Generalized Kripke Frames," *Studia Logica*, vol. 84, no. 2, pp. 241–275, 2006.

[35] M. Gehrke, H. Nagahashi, and Y. Venema, "A Sahlqvist theorem for distributive modal logic," *Annals of Pure and Applied Logic*, vol. 131, no. 13, pp. 65 – 102, 2005.

[36] T. French, "Bisimulation Quantifiers for Modal Logics," Ph.D. dissertation, University of Western Australia, 2006.

[37] V. V. Rybakov, "Barwise's information frames and modal logics," *Algebra and Logic*, vol. 41, no. 5, pp. 323–336, 2002.

[38] I. M. Copi, *Introduction to Logic*, 4th ed. MacMillan Company, 1972.

[39] S. A. Kripke, "A Completeness Theorem in Modal Logic," *Journal of Symbolic Logic*, pp. 1–14, 1959.

[40] C.-J. Liau, "A modal logic framework for multi-agent belief fusion," *ACM Trans. Comput. Logic*, vol. 6, no. 1, pp. 124–174, Jan. 2005.

[41] K. Poulsen, "Hacker Disables More Than 100 Cars Remotely," http://www.wired.com/threatlevel/2010/03/hacker-bricks-cars/, 2010, [Online; accessed 3-October-2011].

[42] Toyota, "Safety Connect," http://www.toyota.com/safetyconnect, 2010, [Online; accessed 1-October-2011].

[43] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental Security Analysis of a Modern Automobile," *Security and Privacy, IEEE Symposium on*, vol. 0, pp. 447–462, 2010.

[44] J. Wittbold and D. Johnson, "Information flow in nondeterministic systems," in *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on*, may 1990, pp. 144 –161.

[45] D. E. Bell and L. J. LaPadula, "Computer Security Model: Unified Exposition and MULTICS Interpretation," *Tech. Rep. ESD-TR-75-306*, 1975.

[46] P. Bieber and F. Cuppens, "A Definition of Secure Dependencies using the Logic of Security," in *Computer Security Foundations Workshop IV, 1991. Proceedings*, jun 1991, pp. 2 –11.

[47] ——, "A logical view of secure dependencies," *Journal of Computer Security*, vol. 1, no. 1, pp. 99–129, 1992.

[48] T. M. Chen, "Stuxnet, the real start of cyber warfare?" *Network, IEEE*, vol. 24, no. 6, pp. 2–3, 2010.

[49] Chen, Thomas M and Abu-Nimeh, Saeed, "Lessons from Stuxnet," *Computer*, vol. 44, no. 4, pp. 91–93, 2011.

[50] M. Clayton, "Stuxnet malware is a weapon out to destroy Iran's Bushehr Nuclear Plant," *Christian Science Monitor*, vol. 21, 2010.

[51] J. P. Farwell and R. Rohozinski, "Stuxnet and the future of cyber war," *Survival*, vol. 53, no. 1, pp. 23–40, 2011.

[52] D. Fidler, "Was Stuxnet an Act of War? Decoding a Cyberattack," *Security Privacy, IEEE*, vol. 9, no. 4, pp. 56–59, 2011.

[53] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, no. 1, p. 13, 2011.

[54] J. Lin, W. Yu, X. Yang, G. Xu, and W. Zhao, "On False Data Injection Attacks Against Distributed Energy Routing in Smart Grid," in *Cyber-Physical Systems (ICCPS), 2012 IEEE/ACM Third International Conference on*. IEEE, 2012, pp. 183–192.

[55] S. McLaughlin, D. Podkuiko, and P. McDaniel, "Energy theft in the advanced metering infrastructure," in *Critical Information Infrastructures Security*. Springer, 2010, pp. 176–187.

[56] Z. Qin, Q. Li, and M.-C. Chuah, "Unidentifiable attacks in electric power systems," in *Proceedings of the 2012 IEEE/ACM Third International Conference on Cyber-Physical Systems*. IEEE Computer Society, 2012, pp. 193–202.

[57] F. P. Preparata, G. Metze, and R. T. Chien, "On the Connection Assignment Problem of Diagnosable Systems," *Electronic Computers, IEEE Transactions on*, vol. EC-16, no. 6, pp. 848–854, 1967.

[58] T. T. Gamage, B. M. McMillin, and T. P. Roth, "Enforcing information flow security properties in cyber-physical systems: A generalized framework based on compensation," in *Computer Software and Applications Conference Workshops (COMPSACW), 2010 IEEE 34th Annual*. IEEE, 2010, pp. 158–163.

[59] T. Gamage, T. Roth, B. McMillin, and M. Crow, "Mitigating Event Confidentiality Violations in Smart Grids: An Information Flow Security-Based Approach," *Smart Grid, IEEE Transactions on*, vol. 4, no. 3, pp. 1227–1234, Sept 2013.

[60] S. I. Hayakawa, *Language in Thought and Action*, 4th ed. Houghton Mifflin Harcourt, 1978.

[61] A. Korzybski, *Selections from science and sanity: An introduction to non-aristotelian systems and general semantics*. The International Non-Aristotelian Library Pub. Co., 1954.

[62] ——, *Science and Sanity: An Introduction to non-Aristotelian Systems and General Semantics*, 5th ed. Institute of GS, 1958.

# VITA

Gerry Howser was born in Missouri, United States of America. He received his B.S. in Physics from the University of Missouri-Rolla(UMR) in May 1974. He did graduate work in Surface Physics at UMR before leaving to work for various parts of the Missouri state government in computing services. In 1985, he left state government to work for Lincoln University, eventually rising to be Director of Computing Services. While with Lincoln, Gerry presented seminar talks on Unisys mainframes and the INTERNET which lead to custom modifications of the Unisys operating system, MCPII.

After a long career with the state and Lincoln University, Gerry turned to teaching and industry training. When the last contract ended, Gerry returned to the Missouri University of Science and Technology (MS&T) to work with Sriram Chellappan to obtain his M.S. in Computer Science in 2012. He was appointed Chancellor's Fellow in Computer Science later that year. Gerry worked with his adviser Bruce McMillin on information flow security in cyber-physical systems. He was awarded his PhD in Computer Science from the Missouri University of Science and Technology in May 2014.

Gerry has published conference papers and peer reviewed journal papers in information flow security and other topics, some of which are listed with the references of this dissertation. While working towards his PhD, Gerry has taught lower and upper level undergraduate classes as a teaching assistant.

Gerry was a member of the MORENET Board of Governers. He has been a member of the Institute of Electrical and Electronic Engineers since 2011. He has been a member of the Association for Computing Machines since 2010.