Fall 2015

Distributed state verification in the smart grid using physical attestation

Thomas Patrick Roth

Department: Computer Science

DISTRIBUTED STATE VERIFICATION IN THE SMART GRID USING PHYSICAL ATTESTATION

by

THOMAS PATRICK ROTH

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

2015

Approved by

Bruce McMillin, Advisor
Daniel Tauritz
Jonathan Kimball
Simone Silvestri
Wei Jiang

ABSTRACT

A cyber process in a distributed system can fabricate its internal state in its communications with its peers. These state fabrications can cause other processes in the distributed system to make incorrect control decisions. Cyber-physical systems have a unique advantage in the detection of falsified states because processes typically have observable effects on a shared physical infrastructure. This physical infrastructure acts as a high-integrity message channel that broadcasts changes in individual process states. The objective of this research is to demonstrate that there are cases where physical feedback from the shared infrastructure can be used to detect state fabrications. To that end, this work introduces a distributed security mechanism called physical attestation that detects state fabrications in the future smart grid. Graph theory is used to prove that physical attestation works in general smart grid topologies, and the theory is supported with experimental results obtained from a smart grid test bed.

ACKNOWLEDGMENTS

It's been 8 years since I first enrolled as an undergraduate at the University of Missouri-Rolla, and I want to thank all of the people who influenced this dissertation. I decided to enroll at Rolla thanks to Clayton Price, and was motivated to pursue research by Matt Buechler. Bruce McMillin won me over with his enthusiasm for cyber-physical security, and has my sincerest thanks for the plethora of opportunities he has given to me over the years. I had the great pleasure as an undergraduate to work with Thoshitha Gamage who shaped my future as a researcher and provided the spark that led to this research topic. And I am very grateful to Saint Louis University High School for building the foundations for these 8 years.

Special thanks to Jonathan Kimball, Daniel Tauritz, Simone Silvestri, Wei Jiang, and Sriram Chellappan for their service on my Ph.D. committee. Both Sajal Das and Maggie Cheng have my gratitude for their courses in graph theory and advanced algorithms which were instrumental in my research. Thanks to Dawn Davis and Rhonda Grayson for their years of administrative support. Good luck to my colleagues Stephen Jackson, Li Feng, and Tamal Paul in their future endeavors.

This research was possible due to the financial support of the Missouri S&T Intelligence Systems Center and the Office of Graduate Studies. It was supported in part by the Future Renewable Electric Energy Delivery and Management Systems Center under the grant NSF EEC-0812121.

Much love to my mother Anne and my aunt Mary Ellen Spencer. Best wishes to my brother Mike and my uncle Bill Niemann. In memory of my father John Roth.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Table Page

1. INTRODUCTION

A cyber-physical system integrates a physical infrastructure with cyber computation for improved performance and reliability. Most CPSs are distributed systems in which several cyber processes cooperate to control a set of physical resources. An individual process cannot directly measure the complete system state, and must communicate over some network to share information with its peers. It is vital that the processes share accurate state information with each other to ensure that the distributed system makes the correct control decisions. Failure in a CPS can result in physical consequences such as damage to the machines or harm to the humans involved in the system operation. These consequences can be severe in critical infrastructures such as water distribution, transportation systems, or the electric power grid.

Consider the case of Stuxnet, which resulted in the wide-spread infection of Siemens programmable logic controllers [1]. The controllers infected by Stuxnet attempted to damage centrifuges by causing malicious changes to their rotor speeds. At the same time, Stuxnet sent false state reports which indicated normal rotor speeds back to human operators. One of Stuxnet's goals was for the false state information to trick the operators into making the wrong control decision, namely, keeping the centrifuges running. Stuxnet was possible because the system was not designed to validate the physical measurements produced by the infected controller.

Stuxnet is an example of an attack that occurred in an industrial setting where physical security could have prevented the attack. However, consider a water distribution system which spans a large geographical area, or a smart house which has a large amount of human traffic. Most CPSs have physical components located in unprotected, public environments. CPS security research, in the same vein as wireless sensor networks [2], should anticipate that parts of the physical infrastructure will be compromised.

It is difficult to detect compromised processes in a distributed system. In some cases, such as Stuxnet, the compromised process continues to exhibit expected communication patterns. Even if a compromise can be detected, it might not be possible to determine the exact process that has been compromised. However, a CPS has an advantage over a traditional distributed system in that it contains a physical infrastructure. When a controller makes a control decision, it leaves a physical footprint. The physical infrastructure serves as a ground truth, rooted in the laws of physics, that constrains the extent to which cyber processes can lie without being detected.

The goal of this research is to develop a distributed security mechanism called *physical attestation* that utilizes both the cyber and the physical aspects of a CPS. Cyber processes can utilize the physical infrastructure as a high-integrity message channel to validate the states of their peers. An example smart grid system will be used to demonstrate the potential of this combined cyber-physical approach to security.

1.1. SMART GRID ARCHITECTURE

An example CPS in which state validation is essential is the future smart grid. Smart grid often refers to the Advanced Metering Infrastructure (AMI) in which smart meters are deployed into the existing electric power grid. These smart meters enable two-way communication between the power utility and their customers. The utility uses smart meters to receive power consumption data from the grid to improve predictions about future load. This data allows the utility to make real time control decisions to improve the performance of the electric power grid. Customers can also receive real time power pricing data from the utility to make more informed choices about power consumption. This research, however, will explore a different model of the smart grid in which consumers participate in power generation through renewable energy resources.

The future smart grid proposes the use of distributed cyber intelligence to manage energy resources at the residential level. This differs from the current power infrastructure in that it allows household consumers to participate in power generation, a role traditionally reserved for the power utility. The future smart grid architecture consists of the houses in a neighborhood, each with its own local energy production and consumption, connected by a shared distribution line. Although the distribution line still connects to the power grid, the cyber intelligences at each house run distributed algorithms such as energy management independent of the power utility.

Figure 1.1 shows the structure of a neighborhood within this smart grid model. Each house, or bus, along the distribution line has some local source of generation and load. Renewable energy resources such as solar and wind provide the generation at each bus, while household appliances contribute to the load. A house with more generation than necessary to satisfy its local load is called a supply house, while a house in need of external generation is called a demand house. One goal in this smart grid model is for supply houses to provide their excess power to demand houses to reduce dependence on the power utility. To enable this goal, the houses are connected by some communication network which they use to share state information and perform distributed computations.

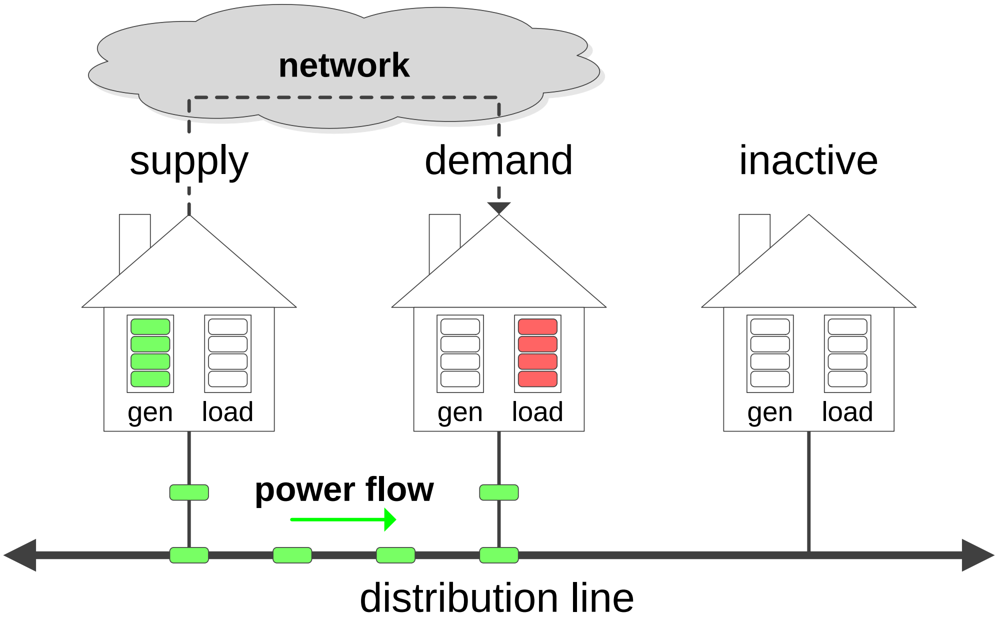


Figure 1.1: Future Smart Grid Architecture

One distributed algorithm for the smart grid is energy management which makes control decisions based on the real power injection at each bus [3]. Each bus has an associated real power injection value that will be used in the distributed energy management algorithm. This power value can be calculated for each bus i as:

$$P_i = Generation_i - Load_i \tag{1.1}$$

The values for $Generation_i$ and $Load_i$ come from the local generation and load at each bus. A negative P_i value indicates that bus i is in demand and acts as a load, while a positive value indicates the bus has supply and acts as a generator. Each demand house can draw power from the distribution line to satisfy its excess load. This capability exists in the current power grid where power flows from a centralized power utility to the individual houses. Supply houses can push some of their excess generation back into the distribution line. This capability represents a supply house using its local generation to satisfy the load at a demand house in its neighborhood.

These pushes and pulls of power are governed by transactions called power migrations that are negotiated between the cyber controllers distributed at each house. Migration contracts are formed to migrate some quanta of power from the supply house into the demand house. A power migration is a sequence of an injection of power from a supply house followed by absorption of power from a demand house. The physical representation of this transaction is that a supplier turns on its generator, causing power to enter the shared distribution line, and then a demand house plugs in some load. This creates a natural flow of power from the supply house to the demand house. Figure 1.2 illustrates the following steps of a power migration:

1. A supply house advertises its excess generation capacity to the neighborhood

2. Demand houses in the neighborhood request power from the supply house

3. The supply house selects a demand house and forms a migration contract

4. The supply house increases its location generation

5. The demand house increases its local load

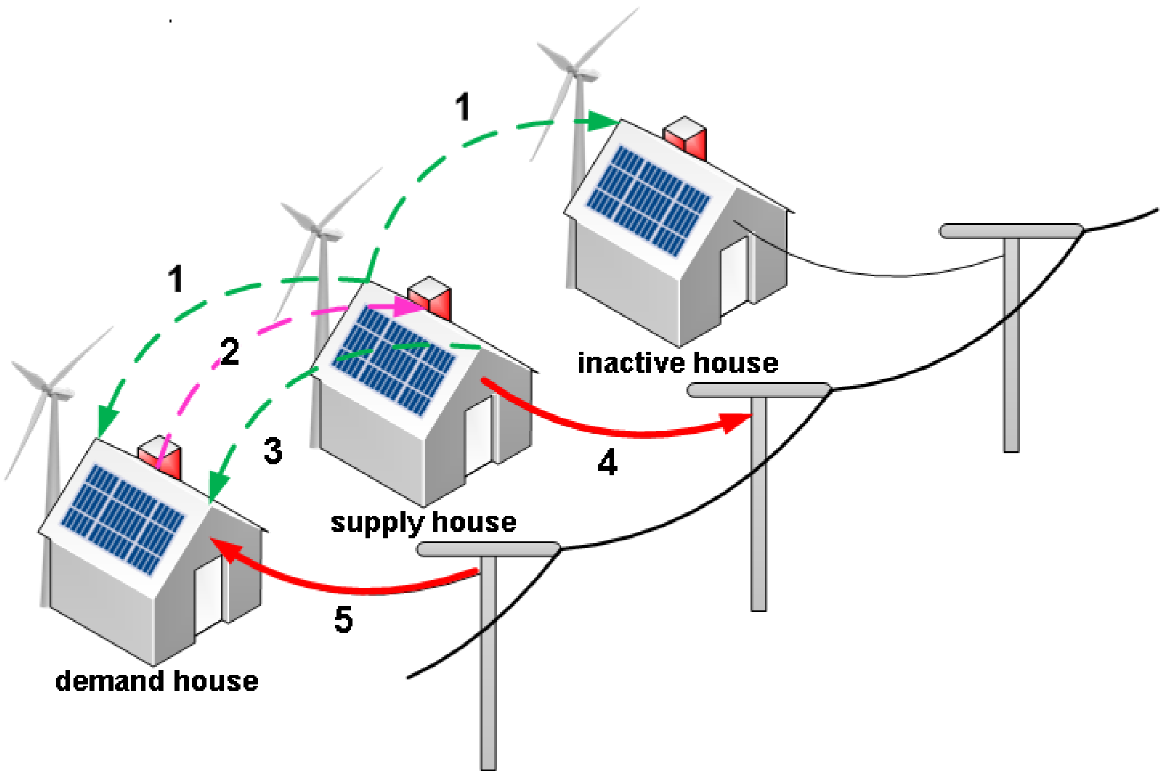


Figure 1.2: Power Migration Steps

A power migration resembles a two-phase commit protocol [4] in that both parties must either perform their respective power changes or abort together. An added difficulty over traditional two-phase commit protocols is that energy management contains both cyber and physical actions. While Step 2 and Step 3 of the migration protocol commit the supply and demand houses to the power migration on the communication layer, they do not guarantee that the corresponding physical actions from Step 4 and Step 5 occur. If a fault or malicious behavior decoupled the sending of the cyber message from the physical action, the power migration would fail.

The formal smart grid model has two state variables to account for this decoupling between the cyber message and the actual state of the physical system. Each bus has two state variables to represent its real power injection, P_i and $\hat{P}_i$. P_i is the amount of excess generation at bus i as calculated by Equation 1.1, and corresponds to the state of the bus in the physical system. $\hat{P}_i$ represents the advertised generation of bus i broadcast to the system during the power migration protocol, and corresponds to the state of the bus in the cyber system. The distinction between these state variables is that P_i is a physical value for the actual generation at a bus, whereas $\hat{P}_i$ is a cyber value sent to other buses over the network. In the case of Stuxnet, P_i would be the actual rotor speed of the centrifuges while $\hat{P}_i$ would be the value for rotor speed that the infected controller reported to the human operator. That the value reported by the cyber controller can be different from the state of the physical system is what enables attacks like Stuxnet.

The buses in the smart grid model are also connected by a shared distribution line which each bus can measure using its local voltage and phase angle. All real power changes that occur, whether from the utility or due to a power migration, will affect each bus' voltage and phase angle. These local measurements allow a bus to gauge whether or not there is activity in the physical system. Each bus i has two additional state variables for voltage V_i and phase angle δ_i to represent these measurements.

1.2. ATTACK MODEL

The smart meters distributed in the AMI are attractive targets for attackers [5]. Smart grid literature has focused on one specific attack scenario against smart meters called the false data injection attack [6]. This is a Stuxnet-like attack in which the state of buses in the power grid is falsified such that it cannot be detected by centralized bad data detection algorithms [7]. It has been shown that false data injection attacks have quantifiable adverse effects on the system, such as increased transmission cost of power and higher outage ratios for households [8]. Although the smart grid model presented in this research differs from the AMI, it is simple to imagine an extension of the false data injection attack in the context of power migrations.

Figure 1.3 illustrates a false state attack against power migrations called the fake demand attack. An attacker compromises a cyber controller associated with one of the buses in the smart grid neighborhood. Like Stuxnet, the compromised controller sends falsified messages over the network which do not correspond to its actual physical state. In the smart grid model, this attack is equivalent to broadcasting a value of $\hat{P}_i$ where $\hat{P}_i \neq P_i$. This allows the attacker, who has no actual load, to pose as a demand node and convince a supplier to give it power that it cannot utilize. It is easy to imagine other similar attack scenarios, such as a case where a supplier lies about generation that it does not have. This work will consider the entire class of false state attacks where $\hat{P}_i$ is falsified, but will use the fake demand attack as a concrete instance for discussion.

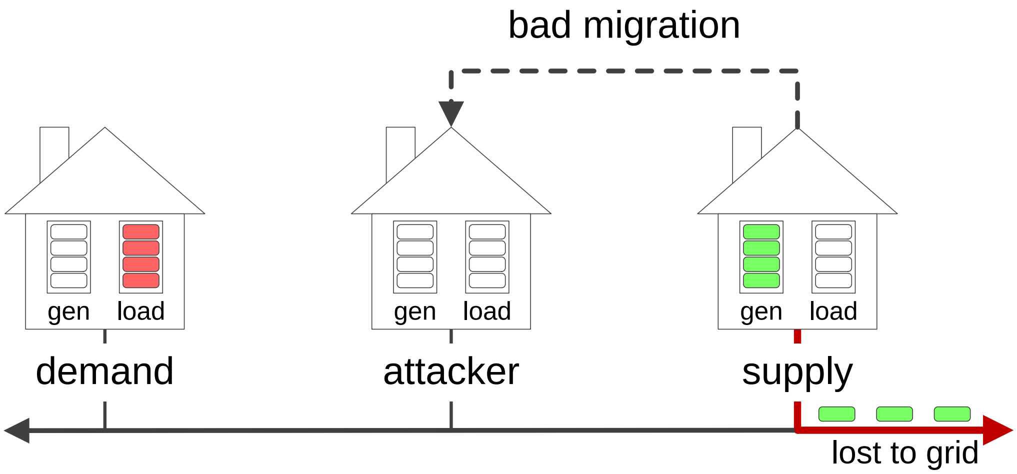


Figure 1.3: Fake Demand Attack

The goal of the fake demand attack depends on whether the system is an isolated micro grid or connected to the wider power grid. If the neighborhood has been isolated from the grid, then the fake demand attack will lead to an imbalance in generation and load that will cause undesired frequency changes [9]. If the neighborhood is grid connected, then the power utility will make up for the difference in generation and load and the frequency will remain stable. However, the attack will prevent the power capacity of the supply houses it tricks from reaching demand houses with an actual need. This will reduce the performance of the power migration algorithm, and can be thought of as a simple denial of service attack against demand nodes in the smart grid.

Suppose a supply house tries to detect this attack by using its local measurements of the shared distribution line to verify the physical behavior of its migration partner. The problem with V_i and δ_i as a means for a cyber process to monitor the behavior of its peers is that the voltage and phase change regardless of where a power change has happened. Consider Figure 1.4 in which the compromised controller A causes a bad power $migration_{BA}$ when it has no local load. There is a change in phase angle δ_B in this case, because although controller A has not changed its amount of load, another demand bus has made a physical change for a separate $migration_{DC}$. From the perspective of the supply house B, it observes activity on the distribution line through a change in its phase angle δ_B and cannot assume that it is the victim of an attack. Although each house has a local measurement it can use to gauge activity on the distribution line, that measurement is not precise enough to identify the sources of power injections. As long as a fake demand attack times itself to occur with other power migrations, an easy feat when the neighborhood contains hundreds of houses, it will not be detected by its peers.

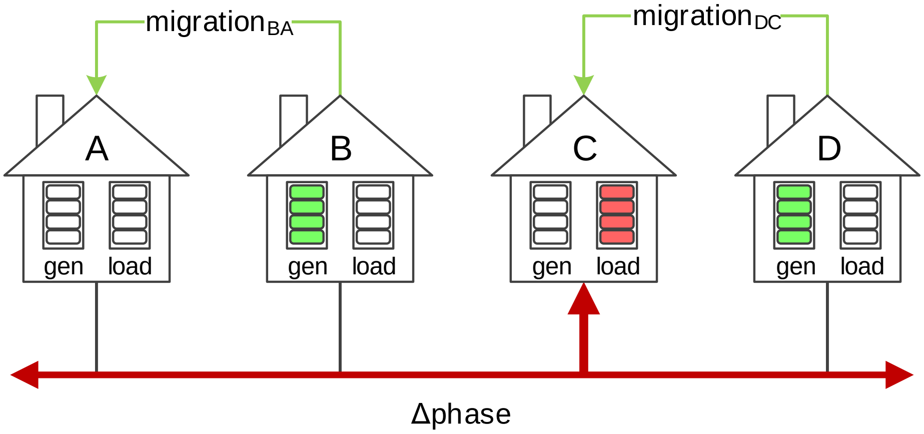


Figure 1.4: Impact of the Fake Demand Attack

This causes a problem for traditional cyber security mechanisms in distributed systems. Both authentication and cryptography do nothing to prevent an insider attack from a compromised controller. Security mechanisms such as anomaly detection and reputation systems depend on the detection of dishonest behavior. Yet from the perspective of the cyber system, no attack has occurred, because the cyber behavior of the compromised controller is consistent with a normal process. The actual attack has occurred through the decoupling of the cyber and the physical actions so that the cyber state no longer corresponds to the physical behavior of a process. What is needed is a means to glue the cyber and physical back together and verify that the reported cyber state of a process corresponds to its actual physical state.

1.3. NONDEDUCIBILITY

In the previous section, the fake demand attack was said to have been undetectable using the local measurements of a single bus in the system. Nondeducibility is an information flow property introduced by Sutherland to analyze system security that can be used to prove this claim [10]. Let a world w be a state of the system. Then define an information function $f_1(w)$ to be all of the high-level events that led to the state w, and a second function $f_2(w)$ to be all of the low-level events. A low-level observation z is nondeducible if:

$$(\forall w \in W)(\exists w' \in W : f_1(w) = f_1(w') \wedge f_2(w') = z \wedge w \neq w') \tag{1.2}$$

Equation 1.2 is satisfied if all possible high-level command sequences $f_1(w)$ can produce the low-level observation z in some world w'. If a single high-level command sequence cannot produce the observation z, then given z it can be deduced that at least one command was not issued, and the Sutherland definition is violated. The traditional use of nondeducibility proves system security by showing that all high-level traces of the system are nondeducible, and so no low-level observation can be used to identify what high-level event occurred. However, the definition is not restricted to the notion of a high and low security domain, and can be extended to encompass any number of different views of the system [11]. Consider two partitions of the system, one partition from the viewport of an attacker (f_1) and a second partition from the viewport of an arbitrary house in the smart grid (f_2). If the attacker is nondeducible with respect to this arbitrary house, it means that the house cannot determine based on its information function whether the attack has occurred. Although nondeducibility is often a desirable property that protects high-level information, in this case a nondeducible system would hide information about an attacker from its peers. Therefore, it is imperative that the system be deducible in order to detect the origin of compromised controllers.

1.4. ATTACK ANALYSIS

A lattice of security levels can be defined for the smart grid that segregates the houses into different security domains. For each bus i in the power grid, a security level L_i exists which represents that bus' perspective of the physical system. An additional *system* level exists to represent the state knowledge shared by all processes in the distributed system, which in this case would be the state messages broadcast over the network. These security levels follow a partial order such that $(\forall i : 1 \leq i \leq n)(system \leq L_i)$. This allows a bus i to view both its private level L_i as well as the shared *system* level, but restricts it from viewing its peers' internal states. Figure 1.5 depicts the security lattice for a system of n nodes.

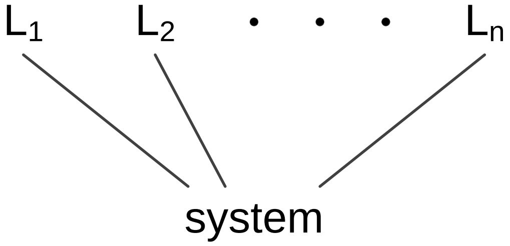


Figure 1.5: System Security Lattice

State variables are assigned to security levels such that, for each bus i in the system, $P_i \in L_i$ and $\hat{P}_i \in system$. Such an assignment has the natural meaning that the amount of generation at a given node is private data known only to that node, and the rest of the system must rely on its view of the advertised value $\hat{P}_i$. This represents the reality of a smart grid where a house is not allowed direct access to their neighbors' meters. $\delta_i \in L_i$ follows the same line of reasoning in that voltage phase angle is unique to each bus, and cannot be read by other buses in the system unless broadcast over the network.

With this, the complete system state Q composed of n buses is defined as the set $Q = \{\hat{P}_1, \ldots, \hat{P}_n, P_1, \ldots, P_n, \delta_1, \ldots, \delta_n\}$. A state transition $Q \rightarrow Q'$ occurs when a node performs a command drawn from a pool of legal system commands. Each command modifies the original state Q to produce a new state Q' where state variables in Q' are represented by prime notation. A list of system commands for power migrations and their impact on the system state are presented in Table 1.1.

Table 1.1: State Transition Commands

command	**description**
migrate(i, j)	supplier i forms a migration contract with demand node j $\hat{P}'_i \leftarrow \hat{P}_i + 1$ and $\hat{P}'_j \leftarrow \hat{P}_j - 1$
increase(i)	node i increases its local generation by one unit $P'_i = P_i + 1$ and $\forall x, \delta'_x = \delta_x + \Delta$ where Δ is some change
decrease(i)	node i increases its local load by one unit $P'_i = P_i - 1$ and $\forall x, \delta'_x = \delta_x + \Delta$ where Δ is some change

Theorem 1. *The fake demand attack is nondeducible to the system if the attacker acts concurrently with another power migration.*

Proof. Table 1.2 shows the command sequence and state transitions for the fake demand attack shown in Figure 1.4. Define the high-level domain as $high = L_A \cup L_C \cup L_D$ and the low-level domain as $low = L_B \cup system$. This definition of the low-level domain is consistent with the view of supply house B which is the victim of the attack. If information flows from *high* to *low*, then supplier B would be able to deduce the high-level commands and thus determine that it has been attacked.

Given the formulation of *high* and *low*, the low-level view consists of states with the form $\{\hat{P}_A, \hat{P}_B, \hat{P}_C, \hat{P}_D, P_B, \delta_B\}$. For the command sequence shown in Table 1.2,

$$\begin{aligned} view_{low} &= \{0,0,0,0,0,0\} \rightarrow \{-1,1,0,0,0,0\} \rightarrow \{-1,1,-1,1,0,0\} \\ &\rightarrow \{-1,1,-1,1,0,\Delta^1\} \rightarrow \{-1,1,-1,1,1,\Delta^2\} \rightarrow \{-1,1,-1,1,1,\Delta^3\} \end{aligned} \tag{1.3}$$

Table 1.2: Fake Demand Attack during Another Migration

cmd	$\hat{P}_A$	P_A	δ_A	$\hat{P}_B$	P_B	δ_B	$\hat{P}_C$	P_C	δ_C	$\hat{P}_D$	P_D	δ_D
init	0	0	0	0	0	0	0	0	0	0	0	0
migrate(B,A)	-1	0	0	1	0	0	0	0	0	0	0	0
migrate(D,C)	-1	0	0	1	0	0	-1	0	0	1	0	0
decrease(C)	-1	0	Δ^1	1	0	Δ^1	-1	-1	Δ^1	1	0	Δ^1
increase(B)	-1	0	Δ^2	1	1	Δ^2	-1	-1	Δ^2	1	0	Δ^2
increase(D)	-1	0	Δ^3	1	1	Δ^3	-1	-1	Δ^3	1	1	Δ^3

Consider the first state transition $\{0, 0, 0, 0, 0, 0\} \rightarrow \{-1, 1, 0, 0, 0, 0\}$ and observe that none of the commands from Table 1.1 can cause this transition except for **migrate**(B, A). It is therefore possible to deduce that this state transition must have been caused by a power migration formed between a supply house B and a demand house A. However, this deduction yields no high-level system information, as the migrate command only affects low-level state variables $\hat{P}_i$. The same is true of the next **migrate**(D, C) command which also operates on only low-level state variables. These two transitions do not help prove the theorem, but provide good examples of what deducible transitions would look like.

The first transition which affects high-level state information is the **decrease**(C) command which causes $\{-1, 1, -1, 1, 0, 0\} \rightarrow \{-1, 1, -1, 1, 0, \Delta^1\}$. If this state transition was dedudcible, then it would be possible to determine which command from the table caused it in the same manner as the **migrate**(B,A) command above. It is immediately obvious that a migrate command did not cause the transition, because none of the $\hat{P}_i$ values have changed. The transition could also not be due to the **increase**(B) command, because the fifth state variable for P_B does not change. However, with the definition of $high = L_A \cup L_C \cup L_D$, these are not high-level commands, as they do not affect variables in the high-level security domain.

There are six possible high-level commands for this transition, the **increase**(i) and **decrease**(i) commands for all i in the set $\{A, C, D\}$. From the command table, all of these commands cause some high-level change to a P_i variable. Yet all of these commands result in the same low-level change of $\delta'_B = \delta_B + \Delta$. Thus, given an initial state of $\{-1, 1, -1, 1, 0, 0\}$, all six commands would lead to the same final state of $\{-1, 1, -1, 1, 0, \Delta^1\}$. This transition is nondeducible since it is possible for all the high-level commands to be issued and still result in the same low-level trace.

Of the remaining two state transitions, **increase**(D) is nondeducible for the same reason as above. While **increase**(B) is deducible since it causes a low-level state change to P_B, this is not high-level state information. Across all the state transitions, only two transitions affect high-level information, and both of those transitions are nondeducible. This makes the entire sequence nondeducible, and from the view of supply house B, it is impossible to determine which subset of houses have made real power changes during the events from Table 1.2. □

Theorem 1 demonstrates that, if an attacker acts during a power migration between another pair of houses, the attacker can pose as a demand node without the *system* being able to determine its identity. Note that the tight timing suggested by this interleaving is not as severe as it seems, as although the attacker must align itself with another power migration, it is difficult to place a causal order between events in a distributed system. Therefore, even if the attacker responds far after the second power migration, it is still possible, given communication delays, for the event sequence to resemble Table 1.2. With this, the fake demand attack has been proven to resemble Stuxnet in that it can hide itself from the view of the system.

1.5. ORGANIZATION

The rest of this document is organized as follows:

1. Section 2 discusses related work in the field of cyber and physical security.
2. Section 3 presents the theory behind a new distributed security mechanism for smart grids called physical attestation.

3. Section 4 describes how physical attestation works in general topologies.

4. Section 5 considers physical attestation in ring, mesh, and hypercube topologies.

5. Section 6 presents preliminary results from an implementation of physical attestation in a smart grid test bed.

6. Section 7 concludes the work with a discussion of contributions and future research directions.

2. RELATED WORK

2.1. TAMPER RESISTANCE

The goal of tamper resistant hardware is to make it difficult to break into the system. For example, the hardware can be enclosed in a sealed box that requires special tools to break open. While tamper resistant hardware cannot prevent a compromise from a persistent attacker, the goal is to raise the cost of the attack. With an increased cost to attack, a reasonable attacker will no longer find it profitable to attack the system [12].

In addition to tamper resistant hardware, there is also tamper responding and tamper evident hardware. Tamper responding means that an attack will trigger some kind of response from the system. For example, a tamper responding car might trigger an alarm when a burglar attempts to break into it. The goal in this case is not to prevent the attack, but to make the attack noticable so an external operator can take corrective measures. Tamper evident means that the attack will leave evidence that can later be recovered to detect the compromise. For example, a tamper evident envelope might have a seal that must be broken before its contents can be read. A tamper evident system makes it easier to apply the other approaches described in this section, as it is easy to determine if a system has been compromised.

There are processors that provide both tamper evident and tamper resistant environments [13]. However, most applications choose to use low-cost processors such as smart cards for improved cost scalability. Research into the effectiveness of tamper resistance in low-cost processors such as smart cards has shown the existence of many viable, low-cost attacks [14]. In addition, because most of these processors involve some form of external communication, software or protocol errors can cause faults that bypass physical security [15].

Because of the cost and limitations of tamper resistant hardware, an equivalent notion of tamper resistant software has appeared in research. A self-checking or integrity-checking piece of software is able to detect modifications to the program code at runtime. This is done through the computation of a checksum of the program execution at runtime, which is compared against original code. If a code modification is detected, the software calls a tamper response mechanism or repair code that restores the modified regions to their original state. Approaches exist that use both insertion of periodic testers into the code [16], as well as insertion of guards on program statements [17].

2.2. BAD DATA DETECTION

Bad data detection is a traditional scheme to protect the state estimation algorithms of the Supervisory Control and Data Acquisition (SCADA) systems that control the electric power grid. Its goal is to determine when a subset of meter readings provided to a centralized state estimation algorithm has been corrupted due to meter faults. The meter readings identified as bad data can be discarded to prevent them from interfering with the control of the power grid.

$$z = Hx + e \tag{2.1}$$

Equation 2.1 shows the basis for state estimation in a power system using a direct current (DC) model for power flow [18]. $z = \{z_1, z_2, \ldots, z_m\}$ is the set of meter readings collected by the SCADA system, and $e = \{e_1, e_2, \ldots, e_m\}$ is the measurement error allowed for each meter. $x = \{x_1, x_2, \ldots, x_n\}$ is the real system state, and H is a vector of functions $h_i(x) = z_i$ that maps the system state to the meter readings z. This equation can be solved for x to get the state estimation $\hat{x}$ based on the known physical topology H and the set of meter readings z.

$$r = z - H\hat{x} \tag{2.2}$$

Equation 2.2 shows an example residual function that can be used for bad data detection [19]. $r = \{r_1, r_2, \ldots, r_m\}$ is a residual that measures the amount of error contained in the estimated state $\hat{x}$. Each element of the residual will be small unless the measurements z contain some amount of bad data. The bad data can be identified by checking against each element if $r_i > e_i$ for some small error tolerance e_i. However, this protection scheme was devised to protect against meter faults, not malicious attacks against the state estimation algorithm.

$$z = Hx + e + a = H(x + c) + e \tag{2.3}$$

Equation 2.3 shows a false data injection attack against the bad data detection algorithm [20]. The a term is an attack vector selected by the attacker which is added to the meter measurements z to inject false data into the system. An intelligent attacker that knows the system topology H will select an attack vector $a = Hc$ to produce the equation $z = H(x + c) + e$. This causes the state estimator to calculate that the system is in some state $\hat{x} = x + c$ other than the actual system state x. Observe that the reason this attack is successful is because the attack $x + c$ is equivalent to some other system state $x' = x + c$ and is therefore nondeducible. Another variant of the false data injection attack is a topological attack which instead modifies H through injection of false topological data [21]. This attack would provide false values of switches and breakers to the state estimation algorithm to produce a state estimate $\hat{x}$ for the wrong physical topology.

Whether false data injection is a threat depends on how difficult it is for an attacker to generate an attack vector a that is consistent with the physical topology. A polynomial time algorithm has been developed that generates the smallest set of meters that must be compromised to launch an unobservable false data injection attack against a given topology [22]. Unobservable in this context means that the attack vector a is consistent with the physical topology $a = Hc$ and will pass through bad data detection algorithms without detection. Another polynomial time algorithm can be used to enumerate all of the low cost unobservable attacks that require 5 or fewer compromised meters in general graphs [23].

The trend in literature to prevent unobservable attacks is to utilize a piece of physical hardware called a phasor measurement unit (PMU). PMUs utilize global positioning system (GPS) synchronized clocks to produce high fidelity meter readings of the power system. It has been shown that deployment of PMUs improves bad data detection against faults due to redundant measurements [24]. It has also been suggested that they can be useful in protection against malicious attacks by providing trusted measurements to the state estimation algorithm [25]. The main assumption behind this approach is that a PMU is tamper resistant, and thus cannot be compromised by an attacker in the same manner as a smart meter. Research in this direction has explored which buses are most vulnerable through formulation of security indices that quantify which buses are most useful in producing unobservable attacks [26]. Redundant measurements can then be produced for these buses through deployment of PMUs to prevent false data injection attacks [27].

Besides tamper resistant approaches, several other methods have been proposed which modify the state estimation algorithms to guard against false data injection. One distributed approach attempts to estimate the false data injection alongside the system state and eliminate the attack vector from the state estimation [28]. Another approach proposes changing the residual from Equation 2.2 to include local measurements of voltage and phase angle at each bus [29]. Observation of which bus voltages lead to high error terms during the state estimation process will isolate the location at which compromises occured during the attack.

2.3. ANOMALY DETECTION

Anomaly detection evaluates the behavior of the nodes in a distributed system for malicious behavioral profiles. If a node deviates from the expected behavior with some statistical significance, or violates an established behavioral rule, then it must be faulty. These two forms of anomaly detection are called profile-based and rule-based detection.

In anomaly detection methods, a subject performs an action on a object which produces an audit record. The actions of all distributed nodes in the system are contained in the same audit record. A real-time system monitors the audit record looking for subjects that exhibit anomalous behavior. Anomalous behavior is defined as a sequence of suspicious actions that have been correlated with an attack. When an anomalous subject is found, the real-time system takes some form of corrective action [30].

A profile-based anomaly detection method first monitors the behavior of all subjects over a trial period. This set of behaviors is used to compute a normal profile for groups of users that is the expected behavior for a normal process. After the trial period, behavior that deviates with statistical significance from the normal behavior is flagged as suspicious. In order to account for changes in the system, the normal profiles must be periodically updated using data from the audit record [31]. A profile-based method of anomaly detection can be replaced with a neural network. Instead of a trial period, the neural network has a training period to adapt to the behavior of the system. Then during runtime, the neural network can map a subject's behavior to one of the trained profiles [32].

As an alternative to profile-based anomaly detection, a rule-based detection method looks for suspicious actions. An intrusion can be modeled as a sequence of signature actions that bring an attack from some initial state to a compromised state. Such an attack can be represented as a series of state transitions, with the transitions being the signature actions. It is possible to develop a set of rules based on the state transitions of known attacks that are able to detect specific attack patterns. Given these attack patterns, the audit record can be parsed for subjects that violate the rules. This approach is better at guarding against known attacks than the statistical approach, but cannot detect new attack scenarios [33].

2.4. SYSTEM LEVEL DIAGNOSIS

System level diagnosis is a method to discover faults in a distributed system by means of peer evaluation. The system is divided into a set of testable units, which are the smallest set of components that can exhibit a fault. Each testable unit is assigned a subset of other units in the system which it is responsible for testing. A test is a challenge-response protocol where a tester issues a challenge to a peer unit and compares the response to some expected value. The set of failed tests in the system is used to determine which units have faults.

The PMC model first introduced the notion of system level diagnosis in literature [34]. A system is modeled as a directed graph where the testable units are vertices and the arcs are test results. Each arc has a label of either 0 for a failed test or 1 for a passed test. The collection of test results is known as a syndrome. There are two possibilities for diagnosis under the assumption that there is some maximum number t of faults in the system. Either all t faults can be identified at once using a single syndrome, or at least one fault can be identified among the t faulted units. These cases are known as one-step diagnosable and sequentially diagnosable systems respectively. A main assumption of the PMC model is that each fault is permanent. This means that a faulted unit will always fail a test issued by a non-faulted unit.

An intermittent fault, or intermittent behavior, permits a fault to either pass or fail when tested by a non-faulted unit. In a system with intermittent faults, the evaluation of testable units may be incorrect due to insufficient testing. A fault in the system can go undetected if the faulted unit exhibits normal behavior during the testing procedure, but faulty behavior otherwise. To detect faults in a system that allows intermittent behavior, it is necessary to use repetitive, periodic testing. Even when the tests are repeated, it is still possible that a faulted unit will result in no failed tests. [35]

A model that considers both permanent and intermittent faults is known as a t/t_i-diagnosable system where t is the maximum number of faults and t_i is the maximum number of intermittent faults. This model is sufficient to represent both the original

PMC model when $t_i = 0$, and the model that considers only intermittent behavior when $t_i = t$ [36].

2.5. REMOTE ATTESTATION

Remote attestation is a mechanism popular in wireless sensor networks which detects compromised sensors using random memory walks. The key assumption behind remote attestation is that faulty behavior leaves some kind of evidence in memory. In essence, remote attestation is one method to implement the tests described in system level diagnosis.

Software-based Attestation (SWATT) attests to the memory contents of an embedded device using an external verifier [37]. It is a challenge-response protocol in which the verifier issues a challenge to a device that instructs it to perform a memory walk. Meanwhile, the verifier performs the same memory walk as a local computation to produce the expected result of the test. When the tested device produces its outcome for the memory walk, it is compared against the expected result to check for faulty behavior. An assumption is made that an attacker cannot modify the physical hardware of the devices, such as memory size or CPU speed. This assumption prevents the attacker from copying the memory contents to external memory for use in the attestation protocol.

Because SWATT is a challenge-response protocol, it is vulnerable to man-in-the-middle attacks. An attacker can intercept the challenge and response to manipulate the outcome of the test results. A better solution is for each device to produce periodic reports that are sent to a verifier without the need for an initial challenge. The synchronized system time can be used as the seed for random number generation in place of a challenge. This protocol, called One-Way Memory Attestation (OMAP), makes remote attestation a more reliable security mechanism. [38]

The SWATT approach also has poor scalability, since a verifier must perform local computation to verify each device. Another revised protocol called Smart Meter Attestation (SMATT) reduces the computational strain by eliminating the need for the verifier to compute an expect result. Instead, the verifier issues a large number of identical, simultaneous challenges to a group of homogenous devices in the system. [39] Because the devices are homogenous, each device should produce the same result in response to the challenge. Therefore, the challenges from each device are compared against each other and responses that differ from the majority are considered malicious. This approach eliminates the need for a verifier to perform a memory walk for each device that must be tested in the system.

A final issue with the SWATT approach is that there may not be a trusted verifier available for each device in the system. In this case, remote attestation can be performed using a cluster of devices as a distributed verifier. Each node in the cluster performs one of the attestation protocols described above independently. A majority vote is then taken between the cluster nodes to determine whether the device passes or fails the test. While other approaches exist to perform distributed attestation, this is the simplest implementation in literature [40].

3. PHYSICAL ATTESTATION

This work proposes a distributed security mechanism for the smart grid called physical attestation. In physical attestation, a verifier validates the measurements produced by a cyber process against the physics of the physical system. Physical attestation is based on the observation that changes in cyber process state cause physical side effects due to the tight coupling between the cyber and physical layers. Prior literature has shown that there are cases where it is possible to deduce the current state of the cyber system by monitoring these physical side effects [41]. The physical infrastructure can be considered a high-integrity message channel that contains a portion of the cyber process state. A verifier can check if a cyber state is consistent with the physical infrastructure to determine whether or not the state has been falsified.

An overview of physical attestation is shown in Figure 3.1. During attestation, a verifier validates the cyber behavior of some target process using a subset of the physical system state. Physical measurements are collected from several processes that neighbor the target in the physical topology to have sufficient observability of the physical system. These measurements are then fed into a physical attestation algorithm which compares the measurements against the topology of the smart grid. Whether or not a target passes attestation depends on whether the measurements used in the verification algorithm are consistent with the physical topology. This section will provide insight on how the physical attestation algorithm is able to detect malicious cyber behavior.

3.1. PHYSICAL INVARIANTS

An invariant is a logical assertion that must remain true during system execution. For a physical system such as the electric power grid, invariants are physical laws that must hold due to the physics of the system. Physical attestation uses a set of physical invariants as the ground truth against which all cyber states are validated. The mea-

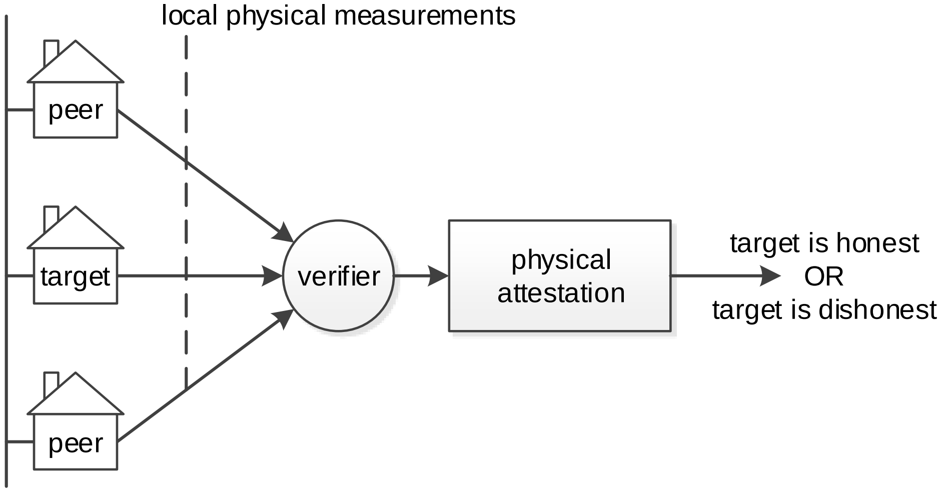


Figure 3.1: Physical Attestation Overview

surements received from the target and its peers are used to instantiate the physical invariants and determine whether the invariants evaluate to true. The theory behind this invariant approach is that a physical law cannot be violated by the physical system. If a physical invariant evaluates to false, then it must be because it was instantiated with bad measurements from a malicious cyber controller.

Within the smart grid, conservation of power can be used as one such physical invariant. In Figure 3.2, for instance, the invariant I_b must hold at bus b that:

$$\{I_b : |P_{ab} + \hat{P}_b - P_{bc}| \leq \epsilon\} \tag{3.1}$$

P_{ij} in Equation 3.1 refers to the real power flow between buses i and j on the distribution line while $\hat{P}_i$ refers to the real power injection value reported by the cyber controller at bus i. Recall that this real power injection value differs from the value in Equation 1.1 because $\hat{P}_i$ is a cyber state that can be falsified by a compromised controller. The ϵ is a small error term to account for measurement error in the physical measurements.

Because the cyber controllers are located at each bus and not each distribution line in the smart grid model, P_{ij} cannot be measured by a single cyber controller. Instead,

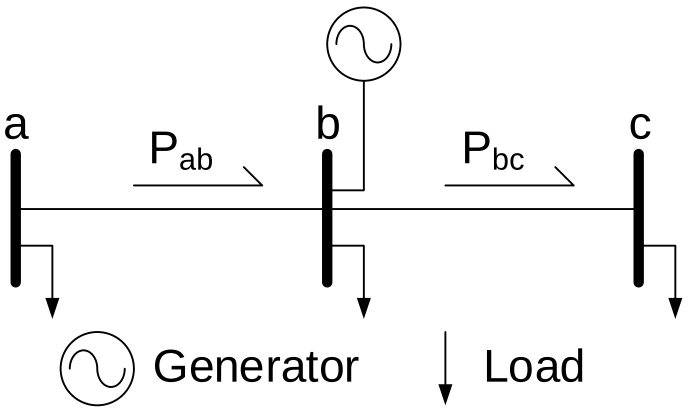


Figure 3.2: Conservation of Power

the line power flows must be calculated using measurements produced by the pair of buses on each side of the distribution line. The real line power flow can be calculated as:

$$P_{ij} = \frac{\hat{V}_i}{R_{ij}^2 + X_{ij}^2}[R_{ij}\{\hat{V}_i - \hat{V}_j cos(\hat{\delta}_i - \hat{\delta}_j)\} + X_{ij}\hat{V}_j sin(\hat{\delta}_i - \hat{\delta}_j)] \tag{3.2}$$

An assumption is made that each bus has a means to measure its local voltage $\hat{V}_i$ and phase angle $\hat{\delta}_i$ such that Equation 3.2 can be instantiated to calculate the distribution line powers for the invariant. A hat notation is used for both of these values because, like the real power injection $\hat{P}_i$, a compromised cyber controller can falsify its reported voltage and phase angle. It is also assumed that each cyber controller knows the topology of the physical system, which includes the distribution line resistance R_{ij} and reactance X_{ij}. Then each bus i will provide its real power injection $\hat{P}_i$ for use in power migrations, and its voltage $\hat{V}_i$ and phase angle $\hat{\delta}_i$ to instantiate Equation 3.2.

3.2. INVARIANT VIOLATION PATTERNS

When a cyber controller feeds falsified values into the system, it will violate a subset of physical invariants. The existence of a violated invariant is enough to deduce the existence of a compromised controller, as a physical invariant can only be violated if instantiated with false cyber states. However, one violated invariant may not be enough to deduce which controller has been compromised, due to the existence of multiple nondeducible cases. This section introduces the notion of invariant violation patterns, and enumerates the conditions that lead to nondeducible violation patterns that hide the identify of the compromised controller.

The electric power grid will be modeled as an undirected graph to simplify reasoning about physical attestation. Let $G = (V, E)$ be an undirected, simple graph where the vertices V are buses and the edges E are distribution lines in the physical system. In order to generate a simple graph, parallel distribution lines in the physical system are combined and represented as a single edge in the undirected graph. Each vertex, or bus, has an associated cyber controller that measures its real power injection, voltage, and phase angle. Each vertex also has an associated conservation of power invariant as defined in Equation 3.1. For convenience, Definition 1 defines a function $\mu()$ that maps a set of vertices into the set of invariants located at those vertices. From this definition, $\mu(\{1, 3, 4\}) = \{I_1, I_3, I_4\}$ refers to the set of invariants located at vertices with the labels 1, 3, and 4.

Definition 1. *$I_x \in \mu(S)$ if and only if $x \subseteq S$.*

Within the system model, each controller can falsify three potential values: $\hat{P}_i$, $\hat{V}_i$, and $\hat{\delta}_i$. However, the voltage and phase angle have the same effect on the invariant as both are used to compute the line power flows in Equation 3.2. It is therefore sufficient to consider all possible permutations of both $\hat{P}_i$ and $\hat{V}_i$ to represent a compromised controller's full ability to attack the system. If an attacker a falsifies state information, then Table 3.1 shows the effect of the attack on the physical invariants. Each attack scenario leads to a specific invariant violation pattern that includes a subset of the compromised controller a and its neighbor set $N(a)$. The neighbor set in this context refers to the buses adjacent to a over the distribution line in the electric power grid.

Table 3.1: Invariant Violation Patterns for a Single Attacker

Falsified State(s)	Violated Invariant(s)
$\hat{P}_a$	$\mu(\{a\})$
$\hat{V}_a$	$\mu(\{a\} \cup N(a))$
$\hat{P}_a\hat{V}_a$	$\mu(N(a))$

When an attacker a falsifies its $\hat{P}_a$ value, it only affects the invariant $\mu(\{a\}) = I_a$ because its real power injection is not required to compute the conservation of power at any other vertex in the graph. However, a falsified $\hat{V}_a$ value is required to compute the line power flows from Equation 3.2 at each neighbor in $N(a)$. These incorrect line flows will then be used in invariants at all of the attacker's neighbors leading to the invariant violations $\mu(N(a))$. An important note is that an attacker cannot violate a proper subset of $N(a)$ unless it provides several different values for $\hat{V}_a$. Since physical attestation has the attacker report $\hat{V}_a$ once to a single verifier process, it will always violate the full set $\mu(N(a))$ of invariants. If a controller falsifies both $\hat{P}_a$ and $\hat{V}_a$, then it can compensate for the incorrect line flows at its own invariant to prevent itself from being violated. This set of observations leads to the violation patterns shown in Table 3.1.

From the set of invariant violation patterns, all nondeducible attacks in the system can be derived. The following discussion will enumerate all possible nondeducible cases for the three attack scenarios in Table 3.1. Throughout the discussion, the same notation as in Table 3.1 will be used to refer to the attack scenarios. Therefore, $\hat{P}_a$ refers to an attack in which vertex a falsifies its advertised real power injection value, while $\hat{V}_a$ refers to a falsifying its bus voltage or phase angle.

Lemma 1. *The false state attack $\hat{P}_a$ is nondeducible if and only if there exists some vertex x where $N(x) = \{a\}$.*

Proof. The attack $\hat{P}_a$ has the violation pattern $\mu(\{a\})$ according to Table 3.1. If $\hat{P}_a$ is nondeducible, then by definition there must exist some other event in the system that causes the same violation pattern of $\mu(\{a\}) = I_a$. Because physical invariants are not violated during normal system operation, this nondeducible event must be an attack scenario caused by some vertex x. For any vertex x in the system, both $\hat{P}_x$ and $\hat{V}_x$ would include a violation at I_x. And because $I_x \notin \mu(\{a\})$, neither these attack can be nondeducible with $\hat{P}_a$. Therefore, $\hat{P}_a$ must be nondeducible with the only remaining attack scenario of $\hat{P}_x\hat{V}_x$. It follows that the $\hat{P}_x\hat{V}_x$ violation pattern $\mu(N(x))$ must be equivalent to $\mu(\{a\})$, which yields the desired result of $N(x) = \{a\}$.

The sufficient proof is obvious from this result. If $N(x) = \{a\}$, then $\hat{P}_x\hat{V}_x$ yields the violation pattern $\mu(\{a\})$ which is nondeducible with $\hat{P}_a$. □

Lemma 2. *The false state attack $\hat{P}_a\hat{V}_a$ is nondeducible if and only if there exists some vertex x where either $N(a) = \{x\}$ or $N(a) = N(x)$.*

Proof. The attack $\hat{P}_a\hat{V}_a$ has a violation pattern of $\mu(N(a))$. In the same manner as the proof for Lemma 1, if $\hat{P}_a\hat{V}_a$ is nondeducible, then there must be another attack by some vertex x that produces the same violation pattern. The attack $\hat{P}_x$ would be nondeducible by Lemma 1 if $N(a) = \{x\}$. It is also trivial to see that $\hat{P}_x\hat{V}_x$ will be nondeducible when $N(a) = N(x)$. The last remaining attack scenario is $\hat{V}_x$. Assume that $\hat{V}_x$ is nondeducible with $\hat{P}_a\hat{V}_a$ in some topology. Then it must hold that these attacks have the same violation patterns, or that $\mu(\{x\} \cup N(x)) = \mu(N(a))$. For this to hold, it must follow that $x \in N(a)$. And because the graph is undirected, $x \in N(a) \rightarrow a \in N(x)$. Given that $a \in N(x)$ and $N(x) \subseteq N(a)$, it must follow that $a \in N(a)$. However, this is impossible because the graph is simple. From this contradiction, $\hat{V}_x$ cannot be a nondeducible scenario. Thus, if $\hat{P}_a\hat{V}_a$ is nondeducible, it must be nondeducible with one of the first two scenarios, which requires either $N(a) = \{x\}$ or $N(a) = N(x)$.

The sufficient proof is once again obvious. If $N(a) = x$, then $\hat{P}_x$ is the nondeducible scenario. If $N(a) = N(x)$, then $\hat{P}_x\hat{V}_x$ is the nondeducible scenario. □

Lemma 3. *The false state attack $\hat{V}_a$ is nondeducible if and only if there exists some vertex x where $\{x\} \cup N(x) = \{a\} \cup N(a)$.*

Proof. The attack $\hat{V}_a$ has the violation pattern $\mu(\{a\} \cup N(a))$. The proofs for Lemma 1 and Lemma 2 have already shown that $\hat{V}_a$ is deducible with both $\hat{P}_x$ and $\hat{P}_x\hat{V}_x$. Therefore, if $\hat{V}_a$ is nondeducible, then it must be nondeducible with the attack $\hat{V}_x$. For this to be true, it must follow that the invariant violation patterns are equivalent and $\mu(\{x\} \cup N(x)) = \mu(\{a\} \cup N(a))$. This leads to the desired result of $\{x\} \cup N(x) = \{a\} \cup N(a)$. $\square$

Lemma 1, Lemma 2, and Lemma 3 list the conditions under which all three attacks from Table 3.1 are nondeducible. There are no other possible attack scenarios in the system model that involve falsified states as this table contains all possible permutations of the three system states. Therefore, if it can be guaranteed that none of the conditions for these three lemma hold, all false state attacks against the system would be deducible. This means that there would be a one-to-one mapping between the invariant violation patterns and the specific attack scenario that caused them. Physical attestation utilizes this result to determine whether a target has been malicious by looking for its unique invariant violation pattern.

3.3. DEDUCIBLE TOPOLOGIES

The three lemma from the previous section provide a means to determine if a given topology is nondeducible with respect to false state attacks involving a single compromised controller. If any of the equalities in the lemma are true, then there is at least one nondeducible false state attack possible in the system topology. However, if none of the equalities hold, then all of the attack scenarios from Table 3.1 are deducible and can be detected from their unique invariant violation patterns. This observation leads to the following theorem.

Theorem 2. *A graph $G = (V, E)$ is deducible with respect to a one-node false state attack if and only if the conditions hold that:*

1. *$\mu(V)$ are invariants*

2. *$\forall x, y \in V, N(x) \neq \{y\}$*

3. *$\forall x, y \in V, N(y) - \{x\} \neq N(x) - \{y\}$*

Proof. Theorem 2 is a direct consequence of Lemma 1, Lemma 2, and Lemma 3. The first condition states the assumption of the system model that there is a conservation of power invariant at each bus in the physical system. The final condition $N(y) - \{x\} \neq N(x) - \{y\}$ is the combination of both $N(y) \neq N(x)$ and $\{x\} \cup N(x) \neq \{y\} \cup N(y)$ into a single statement under the assumption that the graph is simple. Thus, the conditions in the theorem are the conditions in all three lemmas negated to prevent their respective nondeducible attacks. As these three lemma describe all possible one-node false state attacks, these conditions guarantee the absence of nondeducible attacks within the graph. □

Figure 3.3 shows the two topological features that violate Theorem 2. In the first case, x has a degree of 1 with $N(x) = y$ and the second condition of the theorem is violated. This case is nondeducible because there are not enough observation points on x to determine when it has been compromised. A compromised x can implicate y as an attacker, and no one can arbitrate the case because y is the only vertex adjacent to x. The second case, which violates the third theorem condition, shows two vertices x and y with the same neighbor set and an optional edge between x and y themselves. Whether x or y is compromised, the same set of neighbors have invariant violations and be unable to distinguish the source of the compromise. At least one vertex must be connected to either x or y, but not both, to serve as a tie breaker and determine the source of the compromise.

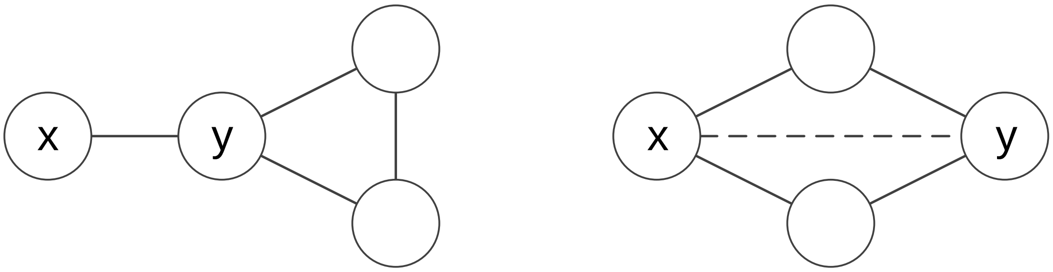


Figure 3.3: Nondeducible Topological Features

If Theorem 2 holds and there are sufficient observation points to detect all of the invariant violations, a compromised controller can be identified by its invariant violation pattern. Consider the physical network shown in Figure 3.4. Through repeated application of the results from Table 3.1 to each of the labeled buses, the set of invariant violations for all possible attacks are presented in Table 3.2. The invariants located below $N1$ and $N5$ are not evaluated because neither $N0$ nor $N6$ exist to provide the voltage and phase angles required to evaluate these invariants. Note that this violates the first condition of Theorem 3.4 in that some of the invariants cannot be evaluated.

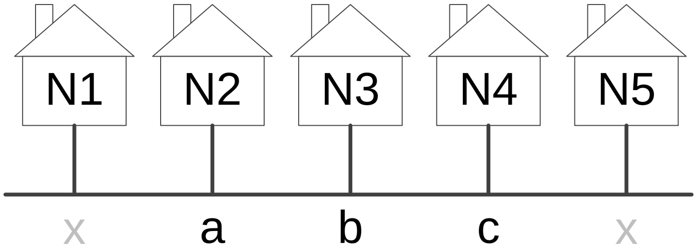


Figure 3.4: Simple 5-Node Topology

Table 3.2: 5-Node Invariant Violations

Malicious Node	**Falsified Value(s)**	**Violated Invariant(s)**
1	$\hat{P}_1$	$\emptyset$
	$\hat{V}_1$	I_a
	$\hat{P}_1\hat{V}_1$	I_a
2	$\hat{P}_2$	I_a
	$\hat{V}_2$	I_aI_b
	$\hat{P}_2\hat{V}_2$	I_b
3	$\hat{P}_3$	I_b
	$\hat{V}_3$	$I_aI_bI_c$
	$\hat{P}_3\hat{V}_3$	I_aI_c
4	$\hat{P}_4$	I_c
	$\hat{V}_4$	I_bI_c
	$\hat{P}_4\hat{V}_4$	I_b
5	$\hat{P}_5$	$\emptyset$
	$\hat{V}_5$	I_c
	$\hat{P}_5\hat{V}_5$	I_c

The invariant violation patterns from Table 3.2 are not unique. I_a can be violated by both $N1$ and $N2$, I_c can be violated by both $N4$ and $N5$, and I_b can be violated by $N2$, $N3$, and $N4$. Observe, however, that the violation pattern for $\hat{V_1}$ does not match the expected results for the behavior of a single compromised controller. This is because two of the invariants that should have been violated fall outside of the observable region of the network. The existence of these border invariants that cannot be evaluated allows for the existence of nondeducible violation patterns. If all of the invariants are calculated, using measurements from all buses in the system, Theorem 2 would be satisfied and there would be no nondeducible cases. However, for a distributed security mechanism such as physical attestation, it is infeasible for one controller to communicate with all of its potentially thousands of peers in the system. Instead, the observability region will be extended by increasing the number of nodes until there are enough observations to generate unique violation patterns for a subset of the controllers.

Figure 3.5 increases the physical network to 7-nodes to extend the observability region of invariant violations. Table 3.3 lists the invariant violation patterns for this physical network. The same problem as the 5-node network exists for the border buses with regards to non-unique violation patterns. For instance, I_{t-2} can still be violated by both nodes $t-3$ and $t-2$. However, node t has a unique violation pattern. No other node violates the combination of invariants I_t, $I_{t-1}I_tI_{t+1}$, or $I_{t-1}I_{t+1}$. This means the 7-node physical network can attest to behavior of node t in the presence of one compromised controller. If the violation patterns match node t, then it is certain that node t is the attacker. Otherwise, it is certain that node t is honest.

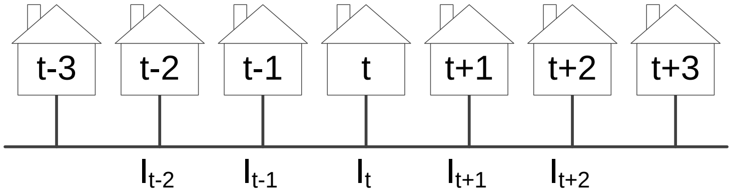


Figure 3.5: Simple 7-Node Topology

Table 3.3: 7-Node Invariant Violations

Malicious Node	**Falsified Value(s)**	**Violated Invariant(s)**
t-3	$\hat{P}_{t-3}$	$\emptyset$
	$\hat{V}_{t-3}$	I_{t-2}
	$\hat{P}_{t-3}\hat{V}_{t-3}$	I_{t-2}
t-2	$\hat{P}_{t-2}$	I_{t-2}
	$\hat{V}_{t-2}$	$I_{t-2}I_{t-1}$
	$\hat{P}_{t-2}\hat{V}_{t-2}$	I_{t-1}
t-1	$\hat{P}_{t-1}$	I_{t-1}
	$\hat{V}_{t-1}$	$I_{t-2}I_{t-1}I_t$
	$\hat{P}_{t-1}\hat{V}_{t-1}$	$I_{t-2}I_t$
t	$\hat{P}_t$	I_t
	$\hat{V}_t$	$I_{t-1}I_tI_{t+1}$
	$\hat{P}_t\hat{V}_t$	$I_{t-1}I_{t+1}$
t+1	$\hat{P}_{t+1}$	I_{t+1}
	$\hat{V}_{t+1}$	$I_tI_{t+1}I_{t+2}$
	$\hat{P}_{t+1}\hat{V}_{t+1}$	I_tI_{t+2}
t+2	$\hat{P}_{t+2}$	I_{t+2}
	$\hat{V}_{t+2}$	$I_{t+1}I_{t+2}$
	$\hat{P}_{t+2}\hat{V}_{t+2}$	I_{t+1}
t+3	$\hat{P}_{t+3}$	$\emptyset$
	$\hat{V}_{t+3}$	I_{t+2}
	$\hat{P}_{t+3}\hat{V}_{t+3}$	I_{t+2}

Table 3.3 cannot be used to detect if nodes other than t are dishonest. Given the invariant violation pattern I_{t-2}, it is not clear whether node $t-3$ or $t-2$ contributed the falsified states that led to the invariant violations. It is therefore necessary to instantiate a different subset of physical invariants dependent on the target of attestation. For example, if the attestation protocol needed to validate the behavior of node $t+1$, then it would shift Table 3.3 one bus to the right and consider observations produced by node $t+4$. One of the main functions of physical attestation is determining which subset of invariants is sufficient to perform attestation without any nondeducible cases. This sufficient invariant set is called an attestation framework, and will be discussed in the next section.

4. ATTESTATION FRAMEWORKS

If a topology satisfies the requirements in Theorem 2, it is guaranteed that there is no single node nondeducible attack in the system. Therefore, the invariant violation pattern maps to a unique attacker when a false state attack occurs. However, this requires measurements from every vertex to be used during the verification process, and does not scale to large systems. Instead, a subset of vertices will be used to perform the verification process, and this subset will be called an attestation framework.

When a subset $S \subseteq V$ of vertices are selected for the attestation framework, it will only be possible to evaluate a subset of the invariants. Definition 2 defines another function $\mu_S(V)$ that selects the subset of invariants from $\mu(V)$ that can be instantiated using measurements from S. From Equation 3.1, an invariant located at a specific vertex cannot be evaluated without measurements from it and all of its neighbors. This leads to the requirements from the definition that both $x \in S$ and $N(x) \subset S$ must hold for the invariant I_x to be evaluated.

Definition 2. *$I_x \in \mu_S(V)$ if and only if $x \in V$, $x \in S$, and $N(x) \subset S$.*

Definition 3. *A subset of vertices S forms an attestation framework for some vertex t if and only if t cannot perform a nondeducible attack using the invariants from the set $\mu_S(V)$.*

Definition 3 implies that the nodes in S will perform a distributed computation to compute a set of invariants using their local measurements. These invariants will then be used to determine if the target t of attestation has performed a false state attack against the system. Because the attestation framework is built around a specific target, each node in the distributed system would have its own associated attestation framework.

4.1. DEDUCIBLE FRAMEWORKS

With the definition of the set of invariants $\mu_S(V)$ and the specific node t, the conditions from Theorem 2 can be changed to consider nondeducibility within a specific attestation framework. First, it is no longer necessary to consider all pair wise combinations of vertices in the system given a specific target t. Now it is sufficient to consider only the cases where either $x = t$ or $y = t$ from the conditions of Theorem 2. Second, the invariants outside of $\mu_S(V)$ cannot be evaluated within a given attestation framework, so all the conditions must be restricted to this specific set. These restrictions lead to the following theorem.

Theorem 3. *A subset of vertices S in a graph $G = (V, E)$ forms an attestation framework for some vertex t if and only if the conditions hold that:*

1. $I_t \in \mu_S(V)$
2. $|\mu_S(N(t))| \geq 2$
3. $\forall x \in N(t) : |\mu_S(N(x))| \geq 2$
4. $\forall x \in V : \mu_S(N(x) - \{t\}) \neq \mu_S(N(t) - \{x\})$

Necessary. Suppose the subset of vertices S forms an attestation framework for t. This proof will show that the removal of any one condition from Theorem 3 enables t to perform a nondeducible attack. The existence of this nondeducible attack would violate Definition 3 and contradict that S is a valid attestation framework.

If the first condition does not hold, then $I_t \notin \mu_S(V)$ and the invariant violation pattern for the attack P_t yields a violation pattern of $\emptyset$. However, $\emptyset$ is consistent with the normal operating behavior of the system in the absence of a false state attack, and thus this attack is nondeducible.

If the second condition does not hold, then $|\mu_S(N(t))|$ must equal either 0 or 1. Suppose $|\mu_S(N(t))| = 0$, then when t performs a P_tV_t attack it results in a violation pattern of $\emptyset$ which is again nondeducible. Suppose $\mu_S(N(t)) = I_x$, then when t performs the same P_tV_t attack it yields a violation pattern of I_x which is nondeducible with P_x.

If the third condition does not hold, then $|\mu_S(N(x))|$ must equal either 0 or 1 for all $x \in N(t)$. However, $x \in N(t)$ combined with $I_t \in \mu_S(V)$ lead to $\mu_S(N(x)) = I_t$ as the only case that can violate this condition. And for this case both P_t and P_xV_x yield the same violation pattern of I_t and are thus nondeducible.

If the fourth condition does not hold, then it is trivial to see that either P_xV_x and P_tV_t or V_x and V_t are nondeducible. Which of these leads to the nondeducible case will depend on whether or not x and t are adjacent. □

Sufficient. Suppose the four conditions from Theorem 3 are satisfied for some subset of vertices S. This proof will show that these conditions lead to t being unable to perform a nondeducible attack using invariants from $\mu_S(V)$. Then Definition 3 leads to the result that S must be an attestation framework for t.

Let t perform the P_t attack with a violation pattern of I_t where $I_t \in \mu_S(V)$ by condition 1 of the theorem. If this attack were nondeducible, then there must exist an $x \in N(t)$ with a violation pattern of $\mu_s(N(x)) = I_t$. However, this would require $|\mu_s(N(x))| = 1$ which violates condition 3, and no such x can exist.

Let t perform the V_t attack with a violation pattern of $\mu(\{t\} \cup N(t))$. This violation pattern within the subset of vertices S is equivalent to $I_t \cup \mu_S(N(t))$. If this attack were nondeducible, then the violation must be due to an $x \in N(t)$ that can violate the I_t invariant. From condition 4, there must be some vertex y where $I_y \in \mu_S(V)$ and either $y \in N(t)$ or $y \in N(x)$, but not both. I_y serves as an indicator that is violated only when its adjacent vertex performs an attack, and its existence makes the attack deducible.

Let t perform the P_tV_t attack with a violation pattern of $\mu(N(t))$. Suppose this attack is nondeducible with some $x \in N(t)$. Condition 2 requires that the P_tV_t attack violate at least two invariants. As a result, x cannot perform the P_x attack which would violate the single invariant I_x. However, all remaining attacks by x would violate $I_t \in$

$\mu_S(N(x))$, and $I_t \notin \mu(N(t))$ gives a contradiction. Suppose this attack is nondeducible with some $x \in V - N(t)$. Then because x and t are not adjacent, $N(t) = N(t) - \{x\}$ and $N(x) = N(x) - \{t\}$. Condition 4 gives $\mu_S(N(x) - \{t\}) \neq \mu_S(N(t) - \{x\})$, which after substitution becomes $\mu_S(N(x)) \neq \mu_S(N(t))$. Therefore, x cannot perform either V_x or P_xV_x as nondeducible attacks, nor the P_x attack as shown in the previous case. As these are all possible attack scenarios, x is deducible with t giving a contradiction. □

Theorem 4. *If Theorem 2 holds for a graph $G = (V, E)$, then the set of all vertices within 4-hops of some vertex $t \in V$ forms a valid attestation framework for t.*

Proof. Let $S = \{t\} \cup h_1 \cup h_2 \cup h_3 \cup h_4$ be the set of vertices within 4-hops of t where h_i contains all the i^{th}-hop vertices. It is sufficient to show that the set S satisfies Theorem 3 to prove that S forms a valid attestation framework. From Definition 2, it follows that $\mu(\{t\} \cup h_1 \cup h_2 \cup h_3) \subseteq \mu_S(V)$ are all framework invariants. Therefore, the first condition of Theorem 3 that $I_t \in \mu_S(V)$ is satisfied.

The condition $\forall x, y \in V, N(x) \neq \{y\}$ from Theorem 2 implies that $|N(t)| \geq 2$. The set $N(t)$ is also the set of 1-hop vertices from t giving the equality $N(t) = h_1$. And all members of h_1 have associated invariants in $\mu_S(V)$ as derived above, which is equivalent to the statement $|\mu_S(h_1)| = |h_1|$. After substitution of these three results, the second condition of Theorem 3 is satisfied that $|\mu_S(N(t))| \geq 2$.

The condition $\forall x, y \in V, N(x) \neq \{y\}$ also implies that $\forall x \in N(t), |N(x)| \geq 2$. If $x \in N(t) = h_1$, then all members of $N(x)$ must come from either $\{t\}$ or the set h_2. From the derivation of $\mu_S(V)$, the set $N(x) \subseteq \{t\} \cup h_2$ are all invariants within the framework and $|N(x)| = |\mu_S(N(x))|$. Substitution back into the first inequality then yields the third condition of Theorem 3 that $\forall x \in N(t) : |\mu_S(N(x))| \geq 2$.

For the final condition, suppose some vertex x exists such that the condition isn't satisfied and $\mu_S(N(x) - \{t\}) = \mu_S(N(t) - \{x\})$. $|\mu_S(N(t))| \geq 2$ has been shown during the course of this proof, which implies both $|\mu_S(N(t) - \{x\})| \geq 1$ and $\mu_S(N(t) - \{x\}) \neq \emptyset$. This result, combined with the original assumption, requires that x and t must have at least one common neighbor. And for x and t to have a common neighbor, it must hold that either $x \in h_1$ or $x \in h_2$. However, this means that all members in $N(x)$ belong to $\{t\} \cup h_1 \cup h_2 \cup h_3$ and must be invariants within the framework. Therefore, the framework invariants associated with $N(t)$ are the same as the invariants outside of the framework and $\mu(N(x) - \{t\}) = \mu(N(t) - \{x\})$. Definition 1 combined with this equality leads to $N(x) - \{t\} = N(t) - \{x\}$ and violates Theorem 2. From this violation, no such vertex x can exist that violates the final condition, and all conditions of Theorem 3 are satisfied. □

Given a random graph that satisfies the topological requirements of Theorem 2, a framework can always be constructed to perform attestation. However, this framework requires a large number of vertices that may not be necessary to perform the actual verification, and will not scale to topologies with high connectivity. A more scalable approach is to minimize the size of the attestation framework to reduce the number of processes that would need to be included in the distributed computation.

4.2. MINIMUM FRAMEWORKS

The minimum attestation framework problem consists of a graph $G = (V, E)$ with a distinguished vertex $t \in V$, and a positive integer $f \leq |V|$. The problem asks: does there exist an attestation framework for t in G with size at most f? An attestation framework consists of a subset of vertices $V' \subseteq V$ that satisfies the topological requirements stated in Theorem 3. This section will prove that the minimum attestation framework problem is NP-Complete using a reduction from set cover.

An instance of the set cover problem consists of a collection C of subsets of a finite set S, and a positive integer $k \leq |C|$. The problem asks: does there exist a subfamily $C' \subseteq C$ where $|C'| \leq k$ and each element of S belongs to at least one member of C'? The set cover problem has a polynomial time solution only for the special case when each set in C has at most two elements.

A polynomial time reduction will transform a set cover instance into a graph G for the minimum attestation framework problem. First, a vertex is created for each set

$C_i \in C$ and each element $s \in S$. An edge exists between the pair of vertices (C_i, s) if $s \in C_i$ in the set cover instance. Then a cycle of 5 new vertices is created that contains the distinguished vertex t. Additional edges are created between t and each element $s \in S$. Given this graph, the value for f is then defined as $f = k + |S| + 5$ where k is the positive integer from the set cover instance. Because each step of this reduction takes either $O(1)$, $O(|S|)$, or $O(|C||S|)$ time, it is a polynomial time reduction.

As an example, consider the set cover instance with $S = \{a, b, c\}$ and $C = \{\{a, b\}, \{a, b, c\}, \{b, c\}\}$. This instance results in the graph shown in Figure 4.1 after the polynomial time reduction. If C does not contain the null set, then this graph will always be connected.

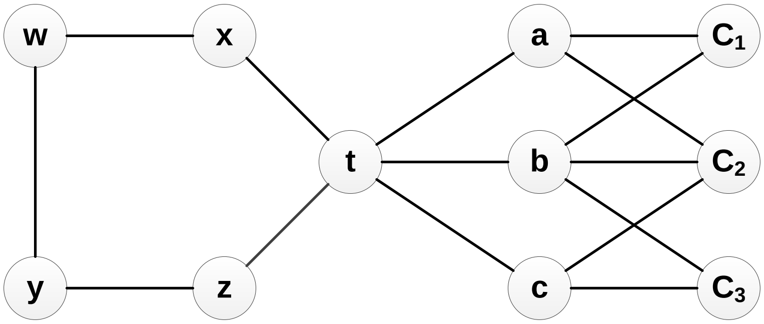


Figure 4.1: Example Reduction from Set Cover

Theorem 5. *There exists an attestation framework for t in $G = (V, E)$ with size $\leq f$ if and only if there exists a set cover $C' \subseteq C$ over S with $|C'| \leq k$.*

Necessary Proof. Suppose there is an attestation framework for t in $G = (V, E)$ with size $\leq f$. Two lemmas will be used to prove the necessary condition of Theorem 5.

Lemma 4. *The vertices from C with associated invariants in $\mu_S(V)$ form a set cover of S.*

The third condition from Theorem 3 requires that each vertex in $N(t)$ be adjacent to two vertices with associated invariants in $\mu_S(V)$. From the construction of G, it holds that $S \subset N(t)$ for all instances of the set cover problem and thus all vertices in S must satisfy this requirement. One of the two vertices will be the target of attestation t since all members of S are adjacent to t and $I_t \in \mu_S(V)$ is guaranteed by the first condition of Theorem 3. From the construction of graph G, the set $\{w, x, y, z\} \cup S$ forms an independent set where $\{w, x, y, z\}$ are the vertices in the 5-cycle with t. The second of the two vertices cannot belong to $\{w, x, y, z\}$ or S due to the existence of this independent set. Therefore, the second vertex must come from the set $V - S - \{w, x, y, z, t\} = C$. Because this requirement must be met by all $s \in S$, it follows that each s must be adjacent to some $C_i \in C$ where $I_{Ci} \in \mu_S(V)$.

Lemma 5. *The number of vertices that correspond to members of C in the framework is at most k.*

Theorem 3 requires that an attestation framework contain the set $N(t)$ in order to evaluate the invariant I_t. Because $S \subset N(t)$, it follows that these $|S|$ nodes must be part of the framework. Now consider the cycle $\{w, x, z, y, t\}$ in Figure 4.1. This cycle will exist in G regardless of the set cover instance that was used in the reduction. For the same reason as above, $\{x, z\} \subset N(t)$ and both vertices must belong to the framework. These two vertices must also be adjacent to a second invariant other than I_t for the framework to be valid. This second invariant must be I_w for x and I_y for z, as both vertices have degree two with their other neighbor being the unusable vertex t. The last vertex in the cycle, t, must be part of the framework since it is the target of attestation. Thus, all vertices in the 5-cycle must belong to the attestation framework.

When these two results are combined, the framework must consist of $|S|$ vertices from S and 5 vertices from the cycle. Because the framework has size at most $f = k + |S| + 5$, there are at most k additional vertices left in the framework. In addition, no vertices from C have been added to the framework as C is disjoint from both the set S and $\{w, x, y, z, t\}$. This completes the proof of Lemma 5.

The necessary condition is a direct consequence of Lemma 4 and Lemma 5: there are at most k members of C in the attestation framework, and these members form a set cover of S. □

Sufficient Proof. Suppose there is a set cover $C' \subseteq C$ over S with $|C'| \leq k$. From the graph G, take the subset of vertices that consists of the solution C' to set cover, the vertices in S, and the vertices in the 5-cycle that contains t. This subset has size at most $f = k + |S| + 5$ which is the desired size of the attestation framework. All that remains is to prove that this subset of vertices forms an attestation framework.

According to Theorem 3, an attestation framework must include an invariant at t. This requires that both t and $N(t)$ be in the framework. For all instances of the set cover problem, $N(t)$ consists of S and two members of the 5-cycle which are included in the subset described above. Because t also belongs to the 5-cycle, it is also included in the subset, and this condition of the theorem is satisfied.

The next condition is that there are two invariants from the set of nodes $N(t)$. These two invariants are the vertices $\{x, z\}$ that are adjacent to t in the 5-cycle. Because the neighbors of these two vertices are members of the cycle, and all members of the cycle are included in the subset described above, the invariants $\mu_S(\{x, z\})$ can be instantiated. This satisfies the second condition of the theorem.

An attestation framework must also include two invariants adjacent to each neighbor of the target t. As I_t can be used as one of these invariants, this requirement is satisfied by finding a set of vertices with invariants excluding t that form a set cover of $N(t)$. The vertices that must be covered are the elements of S and the two vertices $\{x, z\}$ from the 5-cycle. The solution to the set cover problem C' forms a set cover of S. Each of the vertices in the set cover solution C' can also be evaluated as an invariant because their neighbors must come from the set S which belongs to the subset described above. The set $\{w, z\}$ covers the remaining vertices $\{x, z\}$ and are also both invariants since their only neighbors are members of the 5-cycle, all of which belong to the subset described above. This satisfies the third requirement of the theorem.

The last requirement is that no other vertices in the graph are adjacent to the same set of invariants as t. This requirement is trivial through the observation that both I_x and I_z are invariants and $\{x, z\} \subset N(t)$. Through observation of Figure 4.1, no other vertices are adjacent to both x and z, and thus these two vertices will always fulfill the final requirement of Theorem 3.

As each requirement of Theorem 3 has been satisfied with the subset of size $\leq f$ described above, this subset is an attestation framework and the sufficient condition has been proved. □

With Theorem 5 proven, and a polynomial time reduction from all instances of set cover, it has been shown that the minimum attestation framework problem is NP-Hard. And because each of the conditions from Theorem 3 can be verified in polynomial time, the minimum attestation problem is also NP-Complete. From this result, it follows that either an approximation algorithm or special cases must be used in order to generate minimum attestation frameworks.

5. PHYSICAL ATTESTATION IN REGULAR NETWORKS

This section examines several regular networks and shows that, although finding a minimum framework is NP-Hard in the general case, it is quite feasible to find minimum size frameworks for specific network topologies that exhibit regular connectivity.

5.1. RING NETWORKS

This section will derive the minimum size attestation framework for all ring networks. A ring network is a connected topology in which all vertices have a degree of 2. For a large ring with number of vertices $n \geq 7$ it will be shown that the minimum size attestation framework always has a size of $f = 7$. However, this section will also prove the individual cases for rings with size $n < 7$.

Lemma 6. *All ring networks with size $n \leq 4$ are nondeducible with respect to the false state attack and thus cannot be used together with physical attestation.*

Proof. It will be shown that all ring networks that satisfy $n \leq 4$ violate the third condition of Theorem 2, which states that $\forall x, y \in V, N(y) - \{x\} \neq N(x) - \{y\}$. This will be done through showing the existence of a pair of vertices in the graph for which $N(y) - \{x\} = N(x) - \{y\}$. The existence of this counter example is enough to show that Theorem 1 does not apply and the topology is nondeducible with respect to the false state attack.

Consider vertices x and y in the ring network with $n = 2$ in Figure 5.1. For these vertices, $N(x) = \{y\}$ and $N(y) = \{x\}$. When substituted into the conditional above, these two sets lead to $\{x\} - \{x\} = \{y\} - \{y\} = \emptyset$. This completes the counter example which violates the second condition of Theorem 1, and completes the proof for $n = 2$.

Consider the vertices x and y for the network with $n = 3$ in Figure 5.1. $N(x) = \{y, z\}$ and $N(y) = \{x, z\}$ which substitute into the conditional as $\{x, z\} - \{x\} = \{y, z\} -$

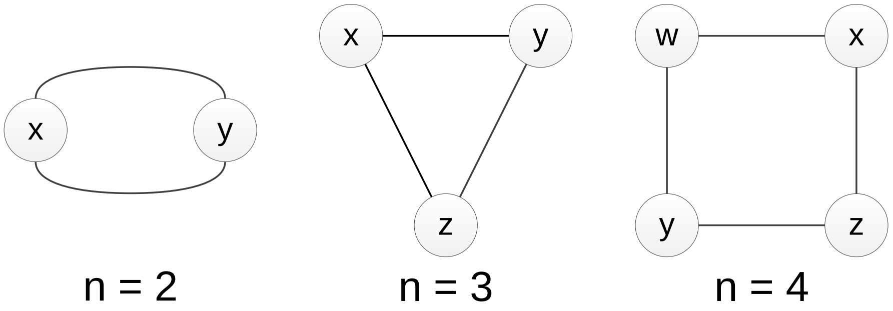


Figure 5.1: Nondeducible Ring Topologies

$\{y\} = \{z\}$. Once again, the successful production of a counter example to Theorem 1 completes the proof.

Consider the vertices x and y for the network with $n = 4$. $N(x) = N(y) = \{w, z\}$. After substitution, it still holds that both $N(y) - \{x\} = N(x) - \{y\} = \{w, z\}$, and this counter example completes the proof for $n = 4$. □

Lemma 7. *All ring networks with size* $5 \leq n \leq 7$ *have minimum attestation frameworks of size* n*.*

Proof. For this proof, an attestation framework will be built around a single target vertex until it satisfies all of the requirements from Theorem 2. It will be shown that this framework requires all n vertices or Theorem 2 will not be satisfied. Due to the unique properties of ring networks, all rings of the same size are homomorphic to each other. Therefore, the minimum framework for one target vertex in the ring can be transformed into the minimum framework for any other vertex by relabeling the nodes in the attestation framework.

For the ring network with $n = 5$ in Figure 5.2, let a be the target of attestation. From Theorem 2, it follows that an invariant must be evaluated at the target a. This requires measurements from the vertices in the set $\{a, b, c\}$ to instantiate the required line power flows P_{ab} and P_{ac}. In addition, there must also be invariants at two neighbors of a to satisfy the second condition of Theorem 2. As b and c are the only possible neighbors of a, both must be invariants, which require measurements from the vertices $\{a, b, d\}$ and $\{a, c, e\}$ respectively. Already, the attestation framework must include all 5 vertices to satisfy Theorem 2 and the minimum framework size for $n = 5$ is n.

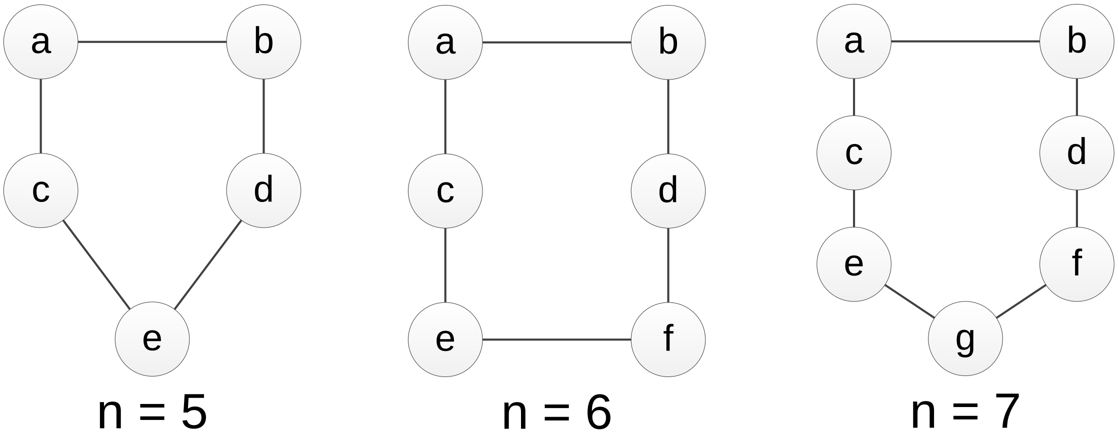


Figure 5.2: Deducible Ring Topologies

Now extend the ring network to $n = 6$ with the addition of the vertex f in Figure 5.2. With a as the target, it still follows that $\{a, b, c, d, e\}$ must be included in the framework for the same reasoning as above. However, the third condition of Theorem 2 states that the neighbors of the target, b and c, must be adjacent to two invariants. For b, this requires an additional invariant at d which will need measurements from the vertices $\{b, d, f\}$ to instantiate its required power flows. This brings f into the framework and increases its total size to 6, and proves the minimum framework size is once again n.

Once again extend the network to $n = 7$ with vertex g in Figure 5.2. It still follows from above that $\{a, b, c, d, e, f\}$ belong to the framework. However, c is also a neighbor of a that must be adjacent to a second invariant. The only option for this second invariant is e, which would require measurements from the vertices in the set $\{c, e, g\}$. As g is not yet in the framework, this increases the total framework size to 7 and shows once again the minimum framework size is n.

This has proved that, in order to satisfy Theorem 2, the attestation frameworks for all ring networks with sizes in the range $5 \leq n \leq 7$ must be of size n. However, it has not shown that attestation is possible in these networks as it was never demonstrated that Theorem 1 holds for these topologies. The method to prove Theorem 1 would be to do an exhaustive proof that there is no pair wise combination of vertices for which the condition $\forall x, y \in V, N(y) - \{x\} \neq N(x) - \{y\}$ is violated like for the cases with $n \leq 4$. This result is obvious because for any selection of two vertices x and y, there always exists a vertex z that is adjacent to either x or y but not both. Due to the mundane nature of this proof, the truth of Theorem 1 is claimed for these three cases without proof. □

Theorem 6. *All ring networks with size $n \geq 8$ have minimum attestation frameworks of size* 7*.*

Proof. Figure 5.3 shows a ring network of size $n > 7$. In the same vein as the previous proof, it will be shown that the minimum size attestation framework for the target a in this network requires a specific number of vertices or Theorem 2 will not be satisfied.

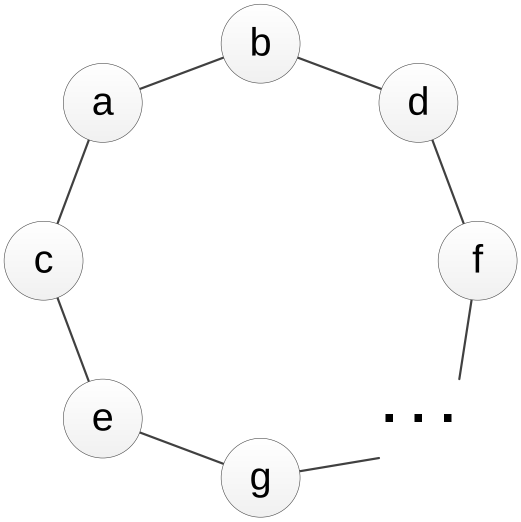


Figure 5.3: Arbitrary Size Ring Topologies

There must be an invariant at a to satisfy the first requirement of Theorem 2. There must also be invariants at both b and c to satisfy the second requirement that at least two neighbors of the target a are invariants. For the third requirement of

Theorem 2, both d and e must also be invariants. The selection of invariants so far follows the same reasoning as the previous proof, and is both sufficient and necessary (due to the limited degree of vertices in ring networks) to fulfill the first three requirements of Theorem 2. The final requirement states that no vertex in the graph can be adjacent to the same set of invariants as the target a. This set is the invariants at both b and c. This final requirement is satisfied since no vertex besides a is adjacent to both b and c. This is obvious through examination of Figure 5.3. Thus, all of the requirements have been satisfied with the set of invariants $\mu_S(\{a, b, c, d, e\})$.

The minimum size framework must be able to calculate these five invariants using Equation 3.1. To do this, it is necessary to calculate the power flows P_{df} and P_{eg}. These power flows require measurements from vertices f and g respectively. The remaining power flows can be computed from the existing set $\{a, b, c, d, e\}$. With this, the vertices that must be included in the minimum size attestation framework are $\{a, b, c, d, e, f, g\}$ with a total size of 7. □

This completes the analysis of physical attestation in ring networks of arbitrary size. Table 5.1 summarizes the results of all of the proofs presented in this section with respect to the minimum attestation framework sizes. The following section will use similar analysis applied to different network topologies.

Table 5.1: Minimum attestation framework sizes for ring topologies

Ring Size (n)	2	3	4	5	6	7	8+
Framework Size (f)	ND[1]	ND	ND	5	6	7	7

5.2. MESH NETWORKS

This section will consider the minimum size attestation framework for a mesh network where each vertex has degree 4. An assumption is made for the sake of generalization that the n-by-m mesh network satisfies $n \geq 6$ and $m \geq 6$. Attestation is

[1] No framework can be formulated as the ring is nondeducible.

possible in smaller networks, but these are special cases which must be considered on an individual basis. In addition, the proof will limit itself to n-by-n mesh networks to utilize graph homomorphism. It will then be argued that additional rows or columns added to this mesh do not affect the size of the minimum framework.

The formulation of the minimum framework will follow in the same manner as the ring network. An arbitrary vertex will be selected to be the target of attestation and vertices will be added one at a time to satisfy all of the requirements of Theorem 3. However, because mesh networks have a higher degree of connectivity, there are often several ways to select vertices to fulfill these requirements. In these cases, multiple frameworks will be constructed that consider each possible permutation of vertices. Due to the regular structure of mesh networks, the actual number of permutations is quite low. All the frameworks will then be compared to determine which has minimum size.

According to Theorem 3, an attestation framework requires at least two invariants in $\mu(N(t))$. Due to homomorphism within a square mesh, there are exactly two unique ways to select these vertices. Either the two vertices appear on opposite sides of the target as shown in Figure 5.4, or the two vertices form an L with the target located at the joint as shown in Figure 5.5. These two variants will be referred to as the line pattern and L pattern respectively. Because these two variants are the only ways to satisfy the degree requirement needed for an attestation framework, the smaller of the two variants will be the minimum size attestation framework for mesh networks.

Lemma 8. *The line pattern results in an attestation framework with total size of 15.*

Proof. Figure 5.6 illustrates each step of the following proof. The line pattern must begin with the invariants in $\mu_S(\{t, 1, 2\})$. I_t is required for the first condition in Theorem 3, while I_1 and I_2 are due to the definition of the line pattern. Another permutation of the line pattern would be $\mu(\{t, a, b\})$, but this is a homomorphic case that will result in the same framework size and can be ignored. The set $\mu_S(\{t, 1, 2\})$ also satisfies condition 4 from the theorem as, by inspection of Figure 5.6, there is no vertex in the graph save t that is adjacent to both vertices 1 and 2. The third requirement is the last requirement

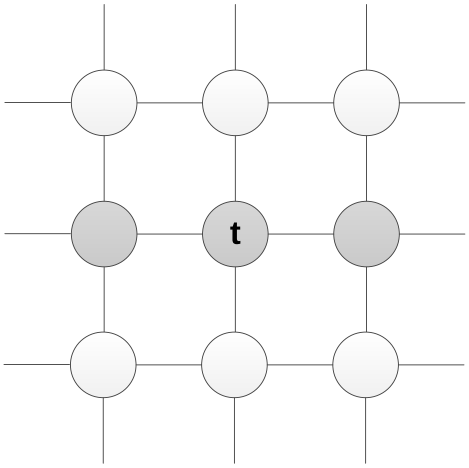


Figure 5.4: Mesh Line Pattern

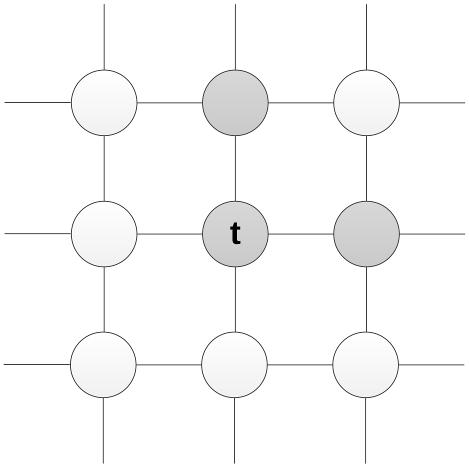


Figure 5.5: Mesh L Pattern

that must be satisfied in order to generate a valid attestation framework. A claim is now made that vertices 3 and 4 belong to a minimum size attestation framework. Under the assumption that this claim holds, then the total framework as shown in Figure 5.6 contains $\mu_S(\{t, 1, 2, 3, 4\})$ and consists of 15 vertices. Recall that several unlabeled vertices must be brought into the framework to satisfy the definition of $\mu_S()$.

If this is not the minimum framework for the line pattern, then it must be possible to replace $\mu_S(\{3, 4\})$ with some other combination of invariants that results in fewer vertices being added to the framework. Notice that it is impossible for condition 3 from Theorem 3 to be met with the addition of a single invariant, as vertices 1 and 2 have no neighbor in common other than the target t and must both be adjacent to a second vertex with an associated invariant in $\mu_S(V)$. Therefore, at least two invariants must be selected for a valid framework. In addition, to be better than the 15 framework described above, at least one of those invariants must add only a single new vertex to the attestation framework. However, there exists no vertex in the graph adjacent to either 1 or 2 that would only increase the framework size by 1. Therefore, the 15 framework described above is the minimum framework that contains the line pattern. □

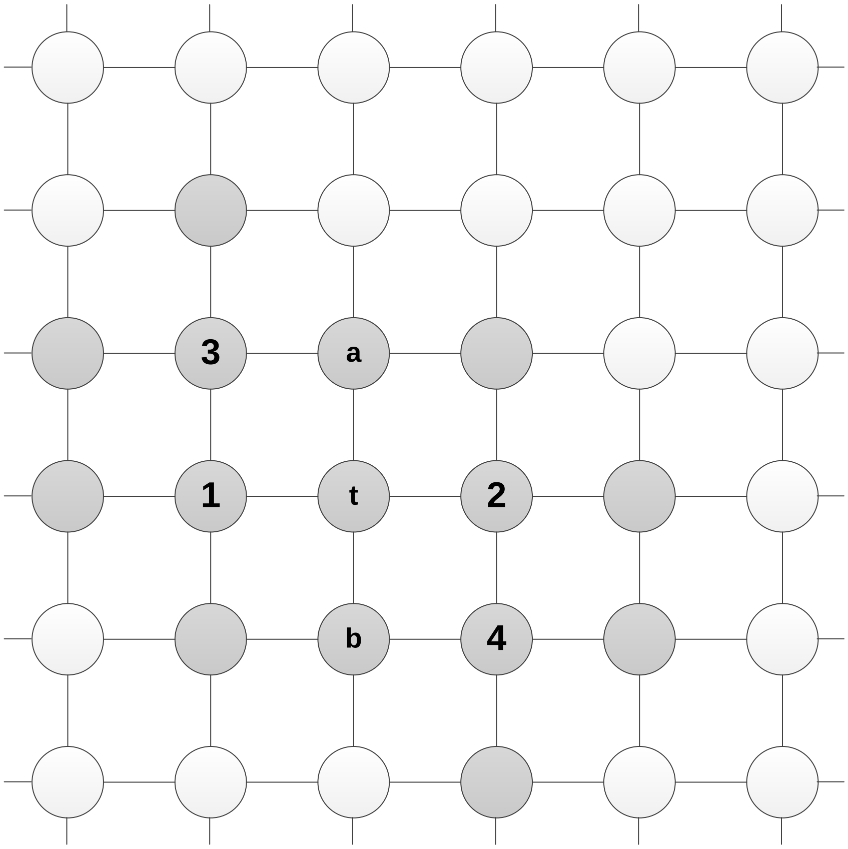


Figure 5.6: 15 Vertex Framework for Line Pattern

Theorem 7. *The minimum size attestation framework for an n-by-n mesh with $n \geq 6$ has size 15.*

Proof. Other than the line pattern covered in Lemma 8, the only other variant for selection of the initial invariants is the L pattern. Figure 5.7 illustrates each step of the following proof. $\mu_S(\{t\})$ must be included to satisfy the first requirement of Theorem 3. Then the L pattern must be included to satisfy the second requirement, which brings in invariants $\mu_S(\{1, 2\})$. In order to satisfy the fourth requirement, another invariant must be added at a vertex that is adjacent to t or a. However, if either I_b or I_c is selected, then the framework would contain the line pattern that was discussed in Lemma 8. Instead, we must chose a new invariant at a vertex adjacent to a. Due to graph homomorphism, the only choice for this invariant is I_3. At this point, the current size of the framework with $\mu_S(\{t, 1, 2, 3\})$ in Figure 5.7 is 13. However, not all of the requirements of Theorem 3 have been satisfied.

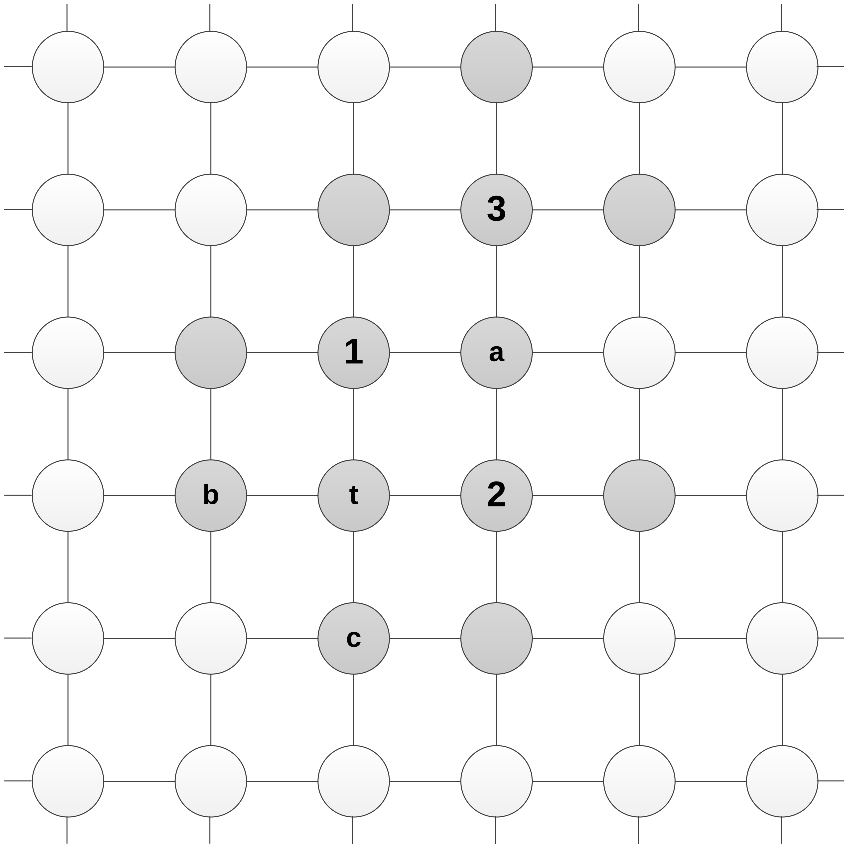


Figure 5.7: 13 Vertex Framework for L Pattern

The last remaining requirement that must be fulfilled is the degree requirement of the vertices adjacent to the target. Consider all the possibilities of invariants adjacent the vertex labeled c in the current framework. All of these possibilities would increase the framework size by at least 2, which would make the minimum framework size no smaller than 15. However, Lemma 8 already describes a framework of size 15. Therefore, using the L pattern, it is impossible to do better than the line pattern, and the line pattern will always result in a minimum attestation framework. □

Corollary 1. *The minimum size attestation framework for an n-by-m mesh with $n, m \geq 6$ has size 15.*

Corollary 1 follows Theorem 7 through the observation that the minimum framework shown in Figure 5.6 did not roll over the edge of the mesh network. Therefore, the addition of rows or columns would not change the proof for Lemma 8 and it would still be possible to generate a framework of size 15 in a rectangular mesh network.

5.3. HYPERCUBES

This section will formulate the minimize size attestation framework for an n-dimensional hypercube. Because of the regular connectivity of a hypercube, construction of this minimum framework can be done in polynomial time. The proof will use the binary formulation of hypercubes where each vertex has an n-bit binary label and two vertices are adjacent if their associated labels have a hamming distance of 1. A special set notation will be used to simplify the discussion in which vertices will be placed into sets based on the number of bits set in their label. These sets will be referred to as h_i for $0 \leq i \leq n$ where i is the number of 1-bits in the label. Figure 5.8 shows an example of a 4-dimensional hypercube and the partition of its vertices into these sets. The minimum attestation framework will be constructed using the single vertex in the set h_0 as the target of attestation. This result can then be generalized using an XOR operation to construct the minimum framework for an arbitrary target node.

Lemma 9. *Some minimum attestation framework for a hypercube uses three invariants from $\mu(h_1)$ to satisfy the second and fourth requirements from Theorem 3.*

Proof. All attestation frameworks must contain two invariants from $\mu(N(t))$ to satisfy the second requirement from Theorem 3. In a hypercube, because the target belongs to h_0 and edges only exist between vertices with a hamming distance of 1, these invariants must be located at vertices in h_1. Given two arbitrary vertices in h_1, there is exactly one vertex in h_2 adjacent to both. Unless additional invariants are added to the framework, this h_2 vertex violates the fourth requirement from Theorem 3. All frameworks must therefore include an additional invariant from $\mu(h_1 \cup h_3)$ to prevent this violation.

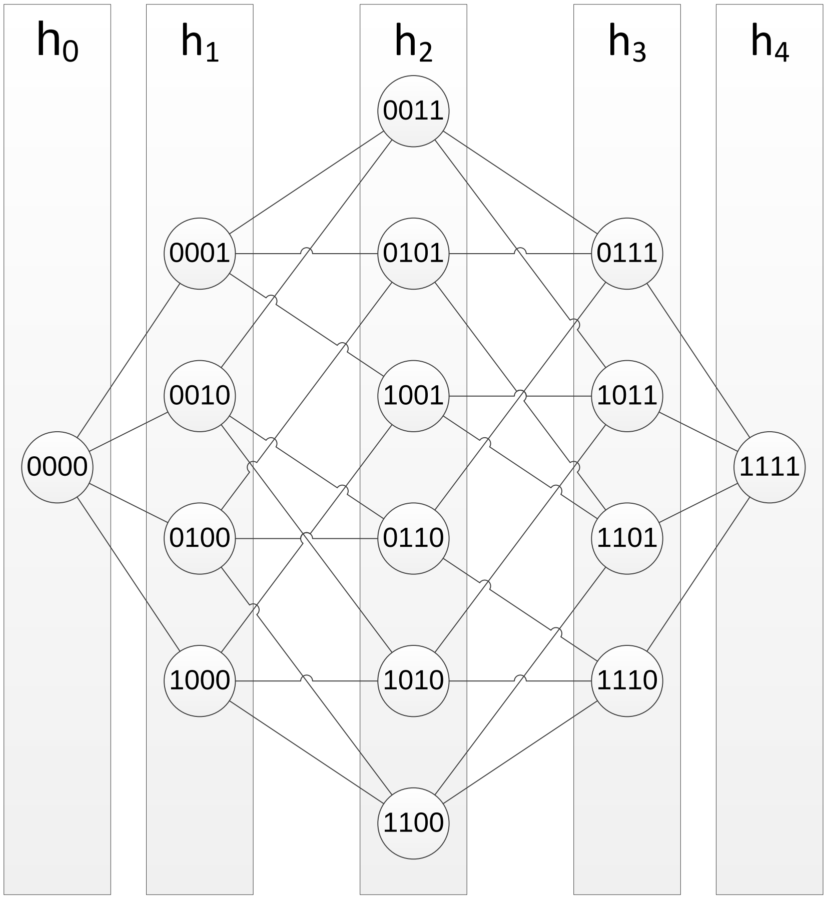


Figure 5.8: Example Hypercube

Assume that a minimum framework uses a vertex $v \in h_3$ to prevent the violation. This vertex will be adjacent to $(n-3)$ vertices from h_4 since it contains $(n-3)$ 0-bits in its label that could be changed into a 1. Until this point, no vertices from h_4 have been added to the framework, and we will now claim that the minimum framework will contain no vertices from h_4 except those $(n-3)$ needed to calculate the I_v invariant being discussed now. Choosing v will therefore always increase the size of the framework $\geq (n-3)$, with the potential for a larger value since additional vertices from h_2 may be needed to calculate I_v.

The proof for the claim that no vertices in h_4 are required outside of this step is derived from Theorem 3. Within the context of a hypercube, the first requirement selects invariants from $\mu(h_0)$, while the second requirement selects invariants from $\mu(h_1)$ and the third requirement from $\mu(h_2)$. This is consistent with the earlier discussion of a two-hop distance being the worst case size for an attestation framework. These invariants would bring vertices from h_0 to h_3 into the attestation framework. No other invariants are required to satisfy Theorem 3 outside of the fourth requirement being considered now, and so the claim that no vertices from h_4 will be required outside of this step holds.

Assume that instead of the h_3 vertex, a minimum framework uses a vertex $u \in h_1$ to prevent the violation of Theorem 3. The h_1 vertex is adjacent to the one vertex in h_0 and $(n-1)$ vertices in h_2. However, h_0 must be in the framework in order to satisfy the first requirement of Theorem 3. In addition, two arbitrary nodes have already been selected from h_1 in order to satisfy the second requirement. Each of these two vertices shares one neighbor in common with u which has already been added to the framework. This means that u would add $\leq (n-3)$ new vertices to the framework.

Because the selection of another vertex from h_1 is always at least as good as selection of a vertex from h_3, there must exist a minimum attestation framework that selects the h_1 vertex. □

From the discussion thus far, we have concluded that some minimum framework contains invariants at h_0 and three vertices from h_1. This framework satisfies the first, second, and fourth requirements of Theorem 3 as discussed in the proof for Lemma 9. The last requirement that must be satisfied is to ensure that each vertex in h_1 is adjacent to at least two vertices with associated invariants in $\mu_S(V)$. As $|\mu(h_0)| = 1$, one of these vertices must necessarily come from the set h_2.

There are two strategies for selection of nodes in h_2 to be invariants: the invariants can be selected either to minimize the number of h_2 vertices added into the framework, or to minimize the number of h_3 vertices. The first approach attempts to minimize the total number of invariants by ensuring that each vertex that is used as an invariant covers as many unique members of h_1 as possible. In the binary notation for hypercubes, the optimal strategy for this approach would be to choose vertices whose labels share no 1-bits in common. For instance, in a 4-dimensional hypercube, selection of (0011) and (1100) since the intersection of these labels is (0000). The second approach attempts to maximize the number of shared neighbors between the vertices selected as invariants in h_3. The optimal strategy for this approach is to select vertices in h_2 that share bits in common. In the 4-dimensional hypercube, selection of (0011), (0101), and (1001) would be an example of this approach since the intersection of these labels is (0001). As it can be seen, these two approaches are incompatible with each other because one attempts to minimize the overlap of 1-bits while the other attempts to maximize the overlap.

To determine the optimal selection of nodes from h_2, the vertices from h_2 are assigned to sets when they are added to the framework. Each set contains the set of h_2 nodes that share mutual neighbors in h_1. Therefore, within a set, the intersection of the labels of the set members will be a bit string that contains a single 1-bit. And across sets, the intersection of any arbitrary label from each set will result in the bit string that contains all 0-bits. In the two examples given above, the sets are $S_1 = \{0011\}; S_2 = \{1100\}$ and $S_1' = \{0011, 0101, 1001\}$. The first approach of minimizing the overlap of 1-bits would attempt to maximize the number of these sets, while the second approach would attempt to minimize the number of sets.

Each set as defined above covers $m+1$ vertices of h_1 where m is the set size. The first member in the set covers 2 vertices in h_1, and each subsequent member contributes only 1 to the total coverage. We assume that if a vertex had covered 0 additional elements, it would have been dropped rather than added to the set. Also note that the coverage of each set is independent of each other. If two sets have overlap in h_1 such that their total coverage is not the sum of their individual coverage, then the two sets could have been combined into one larger set that still satisfies the definition of sets given above.

Each set has a cost equal to the number of new vertices it brings into the framework. The framework already contains all of the vertices in h_1 in order to calculate

I_t, and so these neighbors add nothing to the cost. However, the framework contains no vertices in h_3. We now make the claim that if two vertices from h_2 are placed into different sets, then they share no neighbors in h_3 in common. For two h_2 nodes to have a h_3 neighbor in common, they must share at least one bit. However, if they share one bit, then they must have been placed into the same set by the set definitions. This contradiction proves the claim.

The cost of each set can now be determined in the number of unique nodes from h_3 brought into the framework. The first vertex added to the set is adjacent to $(n-2)$ vertices from h_3 and adds $(n-2)$ to the total cost. In order to minimize the total cost, each subsequent vertex should share 1-bit in common with each member of the set. Then the cost to bring in the $(k)^{th}$ vertex would be $(n-1-k)$ as it would share one neighbor with each of the $(k-1)$ set members before it. It is possible that a set does not satisfy this property, and each subsequent vertex shares a 1-bit with only some subset of the set members. In this case, the entire set can be replaced with a different set that has the same coverage and the cost described above. However, the total cost also must consider the additional vertices brought in from h_2. We now assume that no vertices from h_2 belong to the framework. This is a false assumption, as several vertices from h_2 are already in the framework to evaluate a subset of $\mu(h_1)$. However, this assumption simplifies the next part of the discussion and can be removed later. Under this assumption, each vertex would add $(n-k)$ cost due to the additional cost of 1 to add itself, and the total cost of the set then generalizes to $\Sigma_1^m(n-i)$.

Lemma 10. *Given k arbitrary sets with sizes $\{m_1, m_2, \ldots, m_k\}$ where $\Sigma_1^k(m_i+1) = n$, it is always possible to generate a single set of size $(n-1)$ that has the same set coverage at lower total cost.*

Proof. According to the cost formula, a set of size m has a cost of $\Sigma_1^m(n-i)$. This summation simplifies to $\frac{m(2n-m-1)}{2}$. The cost formula for the two sets are $\Sigma_{i=1}^k\Sigma_{j=1}^{m_i}(n-j)$ and $\Sigma_{i=1}^{n-1}(n-i)$ respectively. Lemma 10 is equivalent to the mathematical inequality $\Sigma_{i=1}^k\Sigma_{j=1}^{m_i}(n-j) \geq \Sigma_{i=1}^{n-1}(n-i)$. Assume instead that $\Sigma_{i=1}^k\Sigma_{j=1}^{m_i}(n-j) < \Sigma_{i=1}^{n-1}(n-i)$. After several simplifications, this inequality reduces to the form $2m_1m_2 + \ldots + 2m_{k-1}m_k < k^2 - k$. There are a total of $\binom{k}{2} = \frac{k(k-1)}{2}$ terms on the left side of this inequality. Because m_i is the size of a non-empty set, it holds that $\forall i, m_i \geq 1$. From this, it always follows that $2m_im_j \geq 2$. Because there are $\frac{k(k-1)}{2}$ of these terms, the entire left side of the inequality must be $\geq 2\frac{k(k-1)}{2} = k(k-1) = k^2 - k$. This gives the contradiction that completes the proof.

□

Corollary 2. *The optimal strategy for selection of invariants in h_2 is to minimize the number of sets and maximize the overlap of 1-bits between the selected invariants.*

Proof. Lemma 10 leads to this result. Because a combined set has a cost no more than multiple disjoint sets, it follows that there exists a minimum attestation framework in which all the invariants from h_2 belong to a single set according to the set definitions. □

Corollary 2 is not the intuitive result as the optimal solution is not to minimize the number of invariants. This result is part of the reason why the minimum attestation framework is an NP-Complete problem. Instead, the optimal solution is to select a single vertex $v \in h_1$ and use the invariants $\mu(N(v) \cap h_2)$. This gives the final piece required for the minimum attestation framework for hypercubes.

Theorem 8. *The minimum size attestation framework for an n-dimensional hypercube contains $(n^2 + 5n - 8)/2$ vertices.*

Proof. The attestation framework includes invariants at h_0, three nodes of h_1 according to Lemma 9, and $(k-1)$ nodes of h_2 according to Corollary 2. The nodes in h_2 require a total cost of $\Sigma_1^{(n-1)}(n-i) = 0.5n(n-1)$ in terms of h_2 and h_3 nodes. There are an additional $(n+1)$ nodes that must be included from h_0 and h_1 to evaluate the invariant located at the target. One of the h_1 invariants can be evaluated at no extra cost by choosing it to be the one bit all the h_2 invariants share in common. The second h_1 invariant can only overlap with one of the h_2 invariants and thus requires $(n-2)$ additional nodes. Likewise, the third h_2 invariant will overlap with one h_2 invariant and one neighbor of the second invariant for $(n-3)$ nodes. The total cost thus becomes $0.5n(n-1)+(n+1)+(n-2)+(n-3)$ which simplifies to the value given in the theorem. □

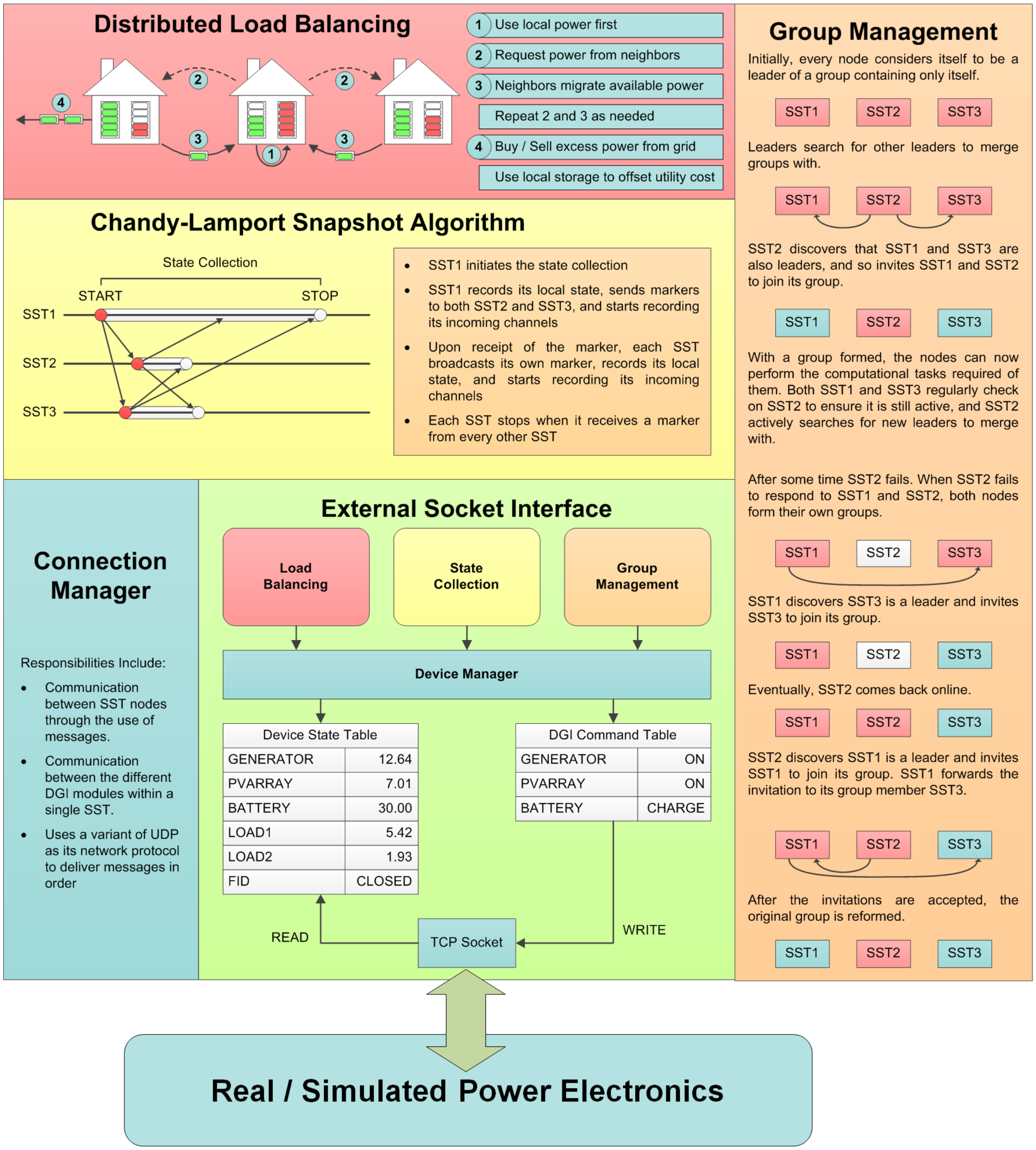
6. EXPERIMENTAL RESULTS

The Future Renewable Electric Energy Delivery and Management (FREEDM) Systems Center is a National Science Foundation funded Engineering Research Center working on the development of an internet for energy [42]. FREEDM has developed simulations of smart distribution networks and a distributed intelligence capable of performing power migrations within the simulation environment. Physical attestation was integrated with the FREEDM system to produce experimental results that support the theoretical results from prior sections. This section provides a brief overview of the implementation details and the experimental results obtained from the smart grid test bed.

6.1. DISTRIBUTED GRID INTELLIGENCE

The Critical Infrastructure Protection Laboratory at Missouri University of Science and Technology has developed the Distributed Grid Intelligence (DGI) to control smart electronics in FREEDM [43]. The DGI provides several services that can be utilized by developers to write distributed algorithms for use in the smart grid. Figure 6.1 shows the main services offered by the DGI.

A Solid State Transformer (SST) is a special piece of hardware developed by the FREEDM Systems Center that enables power resources at a bus to be controlled by the DGI. Within the context of the smart grid model, the SST can be thought of as one of the houses or buses in the distribution grid. Each SST is controlled by a separate instance of the DGI, and the DGI instances communicate with each other using UDP over some network. A service called group management maintains the communication channels between multiple instances of the DGI. The DGI has additional services for collecting the physical state of its attached power electronics, and storing a virtual representation of the physical smart grid topology.


Distributed Load Balancing
1 Use local power first
2 Request power from neighbors
3 Neighbors migrate available power
Repeat 2 and 3 as needed
4 Buy / Sell excess power from grid
Use local storage to offset utility cost
Chandy-Lamport Snapshot Algorithm
State Collection
START
STOP
SST1
SST2
SST3
Group Management
Connection Manager
External Socket Interface
Load Balancing
State Collection
Group Management
Device Manager
READ
TCP Socket
WRITE
Real / Simulated Power Electronics

- SST1 initiates the state collection
- SST1 records its local state, sends markers to both SST2 and SST3, and starts recording its incoming channels
- Upon receipt of the marker, each SST broadcasts its own marker, records its local state, and starts recording its incoming channels
- Each SST stops when it receives a marker from every other SST

Initially, every node considers itself to be a leader of a group containing only itself.

Leaders search for other leaders to merge groups with.

SST2 discovers that SST1 and SST3 are also leaders, and so invites SST1 and SST2 to join its group.

With a group formed, the nodes can now perform the computational tasks required of them. Both SST1 and SST3 regularly check on SST2 to ensure it is still active, and SST2 actively searches for new leaders to merge with.

After some time SST2 fails. When SST2 fails to respond to SST1 and SST2, both nodes form their own groups.

SST1 discovers SST3 is a leader and invites SST3 to join its group.

Eventually, SST2 comes back online.

SST2 discovers SST1 is a leader and invites SST1 to join its group. SST1 forwards the invitation to its group member SST3.

After the invitations are accepted, the original group is reformed.

Responsibilities Include:

- Communication between SST nodes through the use of messages.
- Communication between the different DGI modules within a single SST.
- Uses a variant of UDP as its network protocol to deliver messages in order

Device State Table	
GENERATOR	12.64
PVARRAY	7.01
BATTERY	30.00
LOAD1	5.42
LOAD2	1.93
FID	CLOSED

DGI Command Table	
GENERATOR	ON
PVARRAY	ON
BATTERY	CHARGE

Figure 6.1: Distributed Grid Intelligence Architecture

The most important DGI service for physical attestation is the load balance algorithm that performs power migrations between DGI instances. Each DGI monitors the state of its SST using a device interface which connects to either real or simulated power electronics. It then determines whether it is in the demand or supply state based on its amount of local generation and load. A DGI in the supply state will attempt to form a power migration contract with another DGI in the demand state. These migrations follow the protocol described in the smart grid model introduced in Section 1.

Another DGI service important for physical attestation is device management which handles the physical measurements received from the power electronics. The DGI supports real time measurements from all of the devices attached to its associated SST. As part of the implementation of physical attestation, this service was extended to store historic measurement data during runtime. The historic data is stored with a time stamp that uses the current simulation time of the power simulation. Because all of the DGI instances communicate with the same power simulation, the simulation time acts as a synchronized clock value for the DGI. In a real physical deployment, the simulation time would have to be replaced with an alternate synchronized clock value. Synchronized time is important to ensure that the historic data stored across multiple DGI instances corresponds to a consistent state of the physical system.

The DGI is a real time system, and the various services such as load balance are assigned time slots in a round robin schedule. An example schedule for the DGI is shown in Figure 6.2. One consequence of this schedule is that the different services cannot operate outside of their assigned time slots. All messages that are sent to a particular service outside of its time slot will be delayed until the next time that service is allocated to run. As a result, load balance cannot react immediately to messages from physical attestation when the behavior of a specific target is verified. This delay will appear in the experimental results.

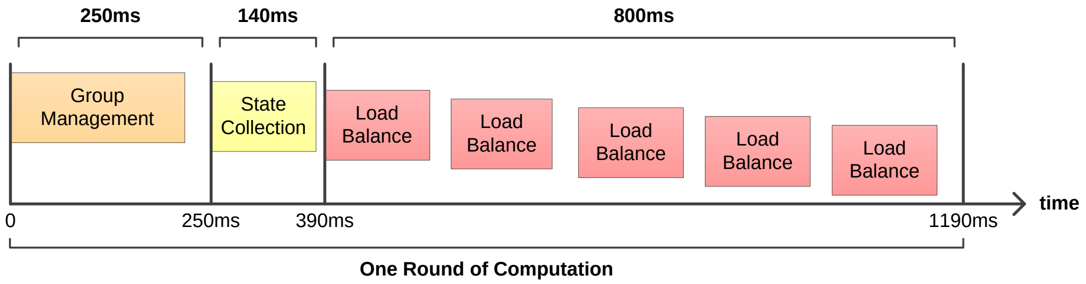


Figure 6.2: Round Robin Schedule

6.2. PHYSICAL ATTESTATION ARCHITECTURE

Physical attestation as a security mechanism determines whether a process in a distribution grid has provided false measurements to some verifier. In the context of power migrations, physical attestation can be used to determine whether a target process has committed to a power migration by making a change to generation or load. If physical attestation fails, it indicates that the power migration has failed and the party in a migration contract with the target should undo their physical action. This leads to a natural situation where both parties in a migration contract will perform physical attestation against each other to verify the physical activity of the other process. Although there are other applications of physical attestation, the following discussion will focus how a supply process in a power migration would use attestation to protect against itself a potential fake demand attack.

Figure 6.3 shows an overview of how physical attestation has been integrated with the DGI. First, a DGI instance must decide to perform attestation on one of its peers. To prevent the fake demand attack, the supply house will serve as the verifier and perform attestation on each demand house it selects for power migrations. Then the supply house must build a valid attestation framework and collect $\hat{P}_i$, $\hat{V}_i$, and $\hat{\delta}_i$ values from all the framework members. These values will be used to compute a subset of the conservation of power invariants to determine the invariant violation pattern. From the violation pattern, the supply house will determine if the demand house it formed a migration contract with is malicious. If the demand house fails attestation, then the supply house must undo its change in generation to back out of the failed power migration.

Physical attestation determines if a process has lied about its physical state at a specific point in time. However, power migrations involve a change in power over time, and thus physical attestation must be performed both before and after the migration was scheduled. A supply house must abort a power migration if its demand partner fails attestation at either of these points, or the real power difference between these two

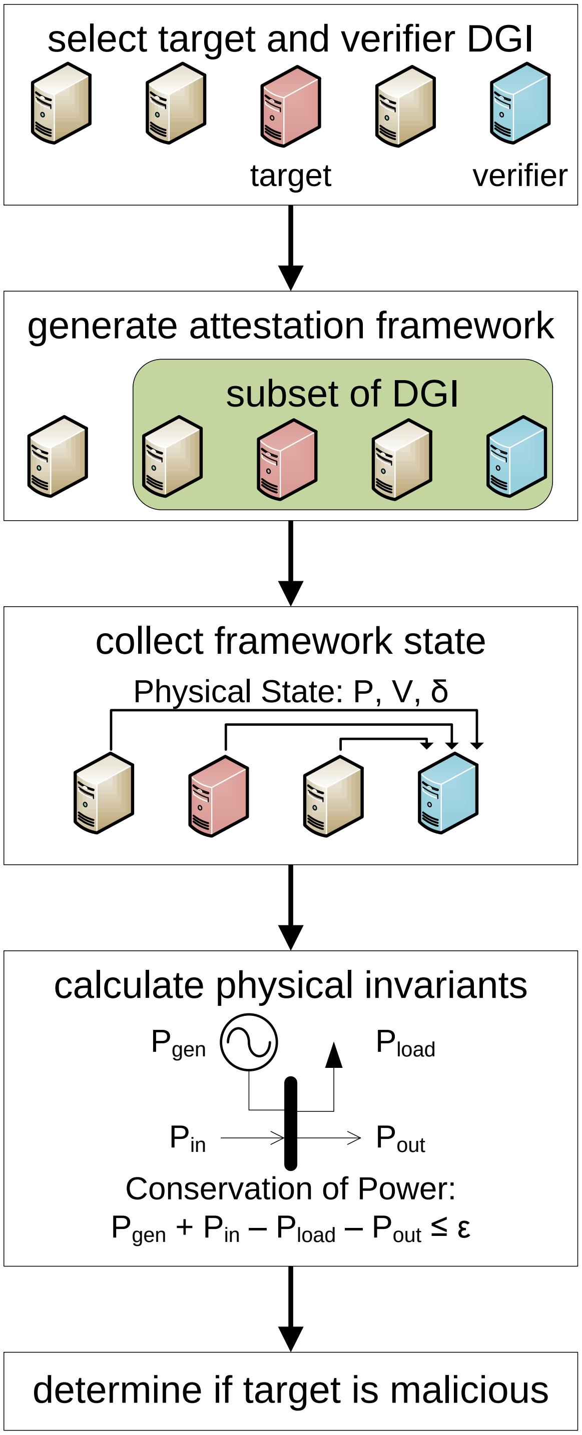

select target and verifier DGI
target
verifier
generate attestation framework
subset of DGI
collect framework state
Physical State: P, V, δ
calculate physical invariants
P_{gen}
P_{load}
P_{in}
P_{out}
Conservation of Power:
$P_{gen} + P_{in} - P_{load} - P_{out} \leq \varepsilon$
determine if target is malicious

Figure 6.3: Physical Attestation in the Distributed Grid Intelligence

points does not correspond to the negotiated migration amount. Algorithm 1 shows how a supply process will call attestation to determine if it should abort a migration due to the existence of a fake demand attack. This algorithm takes as an input the identifier of a target process t, the *time* at which the migration with the target should have occurred, and the real power change ΔP negotiated for the migration. A constant ϵ error term is allowed to account for measurement error, and the expected time τ to finish a migration is used to determine when the change should manifest in the physical system. If the algorithm returns $false$, then either physical attestation reported that the target process has lied or the target did not migrate ΔP units of power at the specified time. A supply process can use the return value of this function to determine whether it needs to undo its change to generation to abort the power migration.

Algorithm 1: Verification of a Power Migration

```
Data: A target process $t$ expected to participate in a power migration
Data: The $time$ when the migration contract was created
Data: The real power change $\Delta P$ expected from the target
Result: A flag to indicate whether the target performed the migration
$\{pass_0, P_0\} \leftarrow$ PhysicalAttestation($t$, $time$)
while $CurrentTime < time + \tau$ do
    $wait$
$\{pass_\tau, P_\tau\} \leftarrow$ PhysicalAttestation($t$, $time + \tau$)
return $pass_0 \wedge pass_\tau \wedge |P_\tau - P_0 - \Delta P| < \epsilon$
```

Algorithm 2 defines the function that performs physical attestation. Line 1 uses a modified breadth-first search algorithm that does not explore past a specified depth, and this generates the 4-hop attestation framework described in Theorem 4. Line 2 checks the framework against the conditions of Theorem 3 to ensure that the framework contains no nondeducible cases for the target t. Line 3 then queries all processes in the framework for their $\hat{P}_i$, $\hat{V}_i$, and $\hat{\delta}_i$ values at the given attestation *time*. These values are fed into an invariant calculation performed on Line 4, and the attestation result is reported back on Line 5.

Algorithm 3 shows the implementation of the check violation pattern function. First the algorithm classifies the framework invariants $\mu_S(V)$ into two sets based on

```
Algorithm 2: Physical Attestation
    Data: The target process $t$ for attestation
    Data: The $time$ when attestation should be performed
    Result: A flag to indicate whether the target passed attestation
    Result: The real power $\hat{P}_t$ reported by the target during attestation
    $S \leftarrow BreadthFirstSearch(t, 4)$
    $pass \leftarrow ValidateFramework(S, t)$
    $data \leftarrow CollectMeasurements(S, time)$
    $pass \leftarrow pass \wedge \neg CheckViolationPattern(S, t, data)$
    return $\{pass, data.t.P\}$
```

whether or not the invariants are located at vertices adjacent to the target t. The first set $I_{adjacent} = \mu_S(N(t))$ contains the invariants adjacent to the target, and the second set $I_{other} = \mu_S(S - N(t) - \{t\})$ contains the remaining framework invariants with the exception of I_t. Then the algorithm counts the number of violated invariants in both sets, and determines whether I_t itself has been violated. This information is used to determine if the violation pattern corresponds to a malicious target t.

In Algorithm 3, the $CalculateInvariant(i, data)$ function is implemented as the conservation of power invariant $|\Sigma_j^{N(i)} P_{ij} - \hat{P}_i| < \epsilon$. The P_{ij} calculation is performed according to the real power flow for a distribution line described in Equation 3.2. However, the power simulation contains three-phase distribution lines, and so Equation 3.2 was computed once for each phase and the results were summed together to get the three-phase power flow.

Theorem 9. *Algorithm 3 returns true if and only if the target t of physical attestation provides false data to the attestation algorithm.*

Sufficient Proof. The key to this theorem is the observation that $I_{adjacent} = \mu_S(N(t))$ and $I_{other} = \mu_S(S - N(t) - \{t\})$. Suppose the target t provides false data to the attestation algorithm. According to Table 3.1, this causes an invariant violation pattern of either $\mu_S(\{t\})$, $\mu_S(N(t))$, or $\mu_S(N(t) \cup \{t\})$. For all three violation patterns, the invariants located at vertices in $S - N(t) - \{t\}$ are true. Therefore, all of the invariants in the set I_{other} are satisfied and $otherViolated = 0$.

Algorithm 3: Invariant Violation Pattern Calculation

```
   Data: The attestation framework $S$
   Data: The target process $t$ for attestation
   Data: A set of $data$ containing $P$, $V$, and $\delta$ values
   Result: A boolean to indicate if the violation pattern matches the target
 1 $I_{adjacent} \leftarrow \emptyset$, $I_{other} \leftarrow \emptyset$
 2 foreach $v \in S - \{t\}$ do
 3     if $\{v\} \cup N(v) \subseteq S$ then
 4         if $v \in N(t)$ then
 5             $I_{adjacent} \leftarrow I_{adjacent} \cup \{v\}$
 6         else
 7             $I_{other} \leftarrow I_{other} \cup \{v\}$
 8 $adjacentViolated \leftarrow 0$
 9 foreach $i \in I_{adjacent}$ do
10     if $CalculateInvariant(i, data) = false$ then
11         $adjacentViolated \leftarrow adjacentViolated + 1$
12 $otherViolated \leftarrow 0$
13 foreach $i \in I_{other}$ do
14     if $CalculateInvariant(i, data) = false$ then
15         $otherViolated \leftarrow otherViolated + 1$
16 $targetViolated \leftarrow \neg CalculateInvariant(t, data)$
17 if $targetViolated \wedge adjacentViolated + otherViolated = 0$ then
18     return true
19 else if $adjacentViolated = |I_{adjacent}| \wedge otherViolated = 0$ then
20     return true
21 else
22     return false
```

If the violation pattern is $\mu_S(\{t\}) = I_t$, then the algorithm will compute that $targetViolated = true$. In addition, the invariants in the set $I_{adjacent}$ must be satisfied which leads to $adjacentViolated = 0$. It follows from substitution that $adjacentViolated + otherViolated = 0$. This causes Line 17 to evaluate to true, and the algorithm returns true.

If the violation pattern is $\mu_S(N(t))$ or $\mu_S(N(t) \cup \{t\})$, then all the invariants in the set $I_{adjacent}$ will be violated and the algorithm will compute $adjacentViolated = |I_{adjacent}|$. Regardless of the value for $targetViolated$, Line 19 $adjacentViolated =$

$|I_{adjacent}| \wedge otherViolated = 0$ will evaluate to true. This causes the algorithm to return true, and proves that all false data provided by the target leads to a return value of true. □

Necessary Proof. If Algorithm 3 returns true, then either Line 17 or Line 19 must evaluate to true. Suppose $targetViolated \wedge adjacentViolated + otherViolated = 0$ is the true clause, then by the formation of both *adjacentViolated* and *otherViolated* it follows that all of the invariants in $I_{adjacent}$ and I_{other} are satisfied. This leads to there being no invariant violations in the set $\mu_S(N(t)) \cup \mu_S(S - N(t) - \{t\}) = \mu_S(S - \{t\})$. The remaining invariant I_t must be false due to the value of the *targetViolated* variable. The invariant violation pattern would then be $\mu_S(\{t\})$, which is consistent with the target having provided the false data.

Suppose the second clause $adjacentViolated = |I_{adjacent}| \wedge otherViolated = 0$ is true instead. Then all the invariants in $\mu_S(N(t))$ must be violated, and none of the invariants in $\mu_S(S - N(t) - \{t\})$ can be violated. The remaining invariant I_t can be either true or false without affecting the truth value of the clause. This leads to the invariant violation pattern of either $\mu_S(N(t))$ or $\mu_S(N(t) \cup \{t\})$, both of which are consistent with the target providing the false data. □

The implementation of attestation described so far detects dishonest behavior from the target due to Theorem 9. However, the existence of a small ϵ term to account for measurement error allows processes to be dishonest within some small tolerance. As such, a process could still shave small amounts of power off of its migrations without being detected by Algorithm 1. In addition, the τ term for the time it takes the system to complete a power migration introduces a small delay to the load balance algorithm. All power migrations must be separated by at least τ units of time, as otherwise attestation will result in false positives when multiple migrations happen within one time period.

6.3. SMART GRID TEST BED

The Center for Advanced Power Systems (CAPS) at Florida State University manages a Hardware-in-the-Loop (HIL) simulation test bed for the smart grid [44]. A HIL simulation allows for integration of multiple DGI instances with a real time power simulation to control the simulated power electronics. Figure 6.4 shows a high level overview of the test bed architecture.

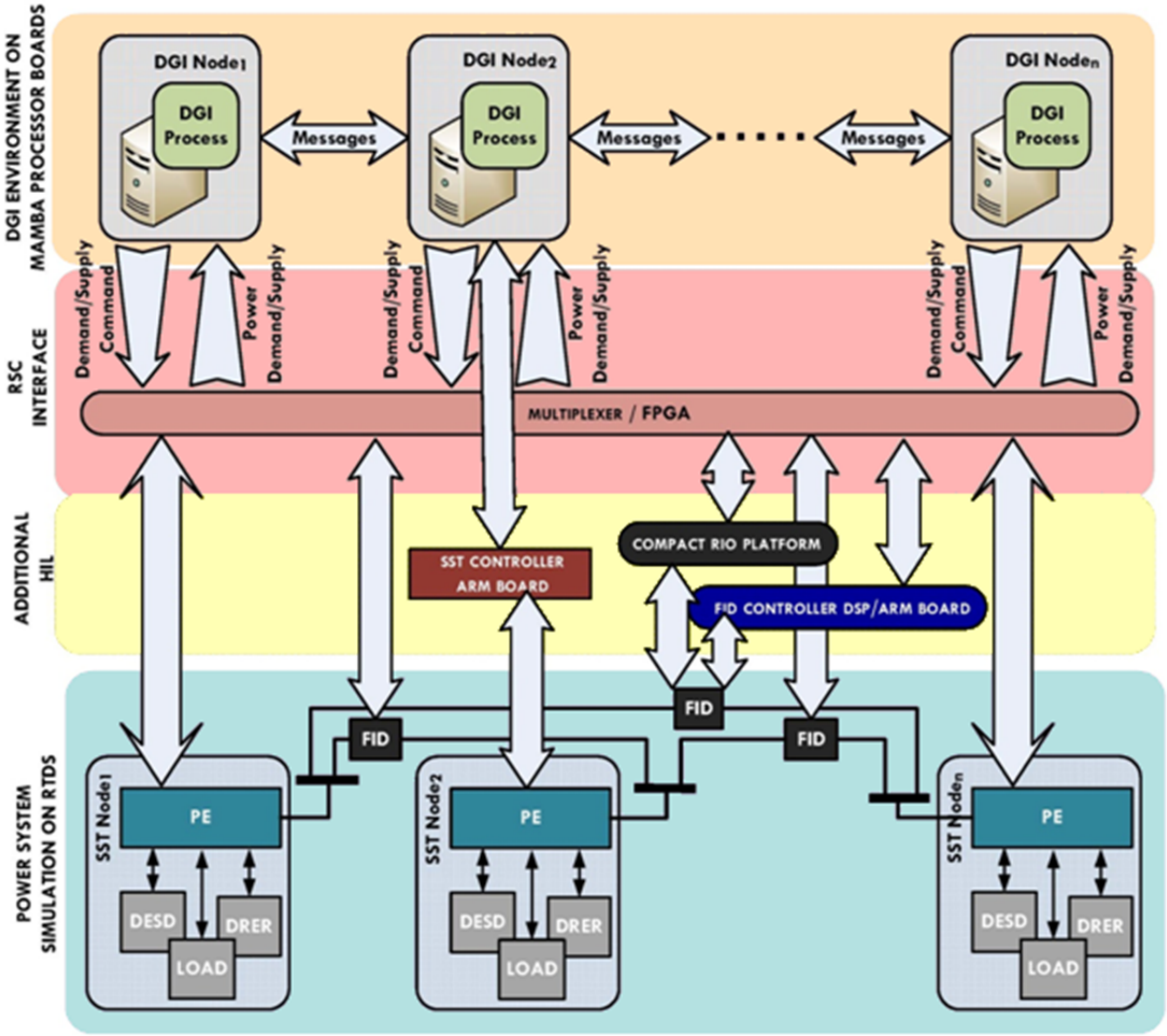


Figure 6.4: Hardware-in-the-Loop Simulation Architecture

The test bed contains six Mamba boards and one field-programmable gate array (FPGA) connected via Ethernet to a local area network. DGI instances are deployed across the six Mamba boards, with up to four simultaneous DGI instances per board.

The DGI communicate with each other using UDP through the wired internal network, and with the power simulation using a TCP connection to the FPGA. The FPGA maintains a separate TCP socket for each DGI instance, and acts as a multiplexer that pushes the DGI's commands to the power simulation. Metered values from the simulation also go through the FPGA where they are sent to the DGI instance that requires that specific state information.

A power simulation is compiled and run on a real time digital simulator (RTDS). The values sent to and received from the FPGA must be specified before at compile time. An additional Windows desktop computer not shown in the architecture is connected to the RTDS to monitor its runtime. Through this Windows machine, parameters of the simulation can be monitored and changed while the simulation is running.

During a typical simulation, the RTDS will start in a steady state where the DGI is idle. A command will be issued through the Windows machine to perturb the power system and cause a reaction in the DGI. The simulation will be left alone until the power system enters its next steady state, and the resulting graph will be used for the experimental results. In the case of the load balance algorithm, the command that will be issued to the simulation is a change in the generation and load at specific buses.

The physical topology used on the test bed is shown in Figure 6.5. The seven numbered buses model SSTs rated at 285 kVA that can be configured to act as either generators or loads. An eighth bus labeled $BusS$ has no local generation and load and acts as the connection point for one of the two diesel generators in the system. Both diesel generators $G1$ and $G2$ are rated at 1.4 MVA, 380 V_{L-N}, with $G1$ operating in isochronous mode and $G2$ operating in droop mode. $G2$ runs at 400 kW nominal with a 5% droop. The generators are connected to the system by a transformers rated at 1.5 MVA, 0.83 kV:12.47 kV. All the distribution lines connecting the buses are identical, with positive sequence resistances of 0.7704Ω and positive sequence inductances of 1.967 mH.

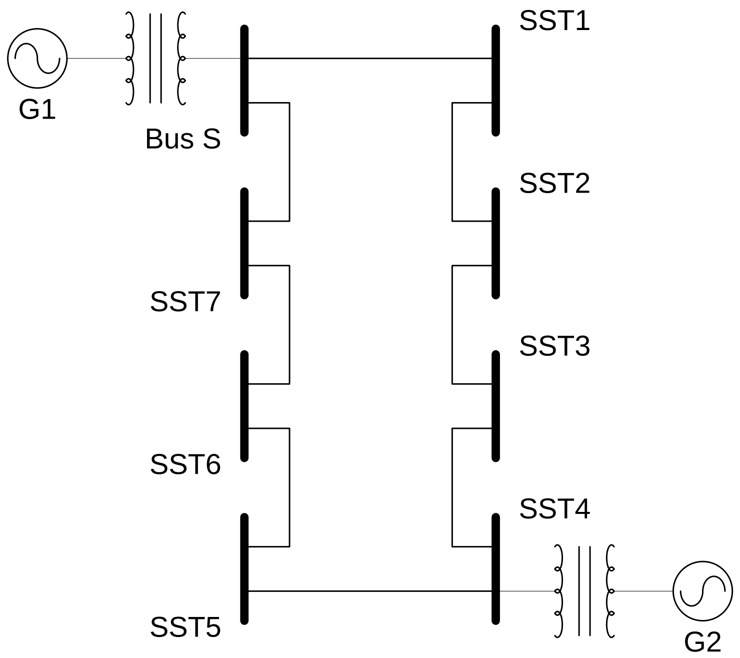


Figure 6.5: Simulated Distribution Grid Topology

6.4. EXPERIMENTAL SETUP AND RESULTS

One limitation of the DGI load balance algorithm is that it does not work well when attached to both fixed and controllable sources of generation. For instance, SST4 contains both its local renewable generation as well as the droop generator $G2$ which produces a constant 400 kW. If the DGI instance at this SST participated in the load balance algorithm, it would attempt to migrate power from the droop generator despite its need to remain constant. SST4 therefore must be split into two buses controlled by different DGI instances to separate the droop generator from the renewable generation. Both DGI will read identical state values from the power simulation, but consider themselves to be on separate buses connected by a 0 impedance distribution line.

An additional hardware limitation is that the FPGA can only transfer 64 data points out of the simulation. However, the simulation requires 9 DGI instances and each instance requires 8 data points. It would be impossible to run the simulation with instances of DGI associated with all of the buses using the current test bed hardware. Therefore, only 7 DGI instances were run on 7 sequential buses in the power system. This allows for the center DGI instance to be a valid attestation target as shown in the 7-node framework from Section 3.

Table 6.1 shows the configuration of the buses for all of the experimental results discussed in this section. Buses 6 and 7, which do not have associated DGI instances due to the hardware limitation discussed above, were set to produce a fixed 250 kW of load. This ensures that there is always sufficient load in the system to allow the droop generation to run at its rated value of 400 kW. Bus 3 acts as the demand house for power migrations and will increase its load from 0 to 285 kW during the simulation runtime. Bus 2 acts as both the supply house and the verifier that will perform physical attestation.

Table 6.1: Experimental Bus Configuration

Bus ID	DGI Instance	Generation	Load
S	mamba1:51870	100 kW	0
1	mamba2:51870	0	0
2	mamba3:51870	0 → 285 kW	0
3	mamba4:51870	0	0 → 285 kW
4a	mamba5:51870	0	0
4b	mamba6:51870	400 kW	0
5	mamba1:51871	0	0
6	*none*	0	250 kW
7	*none*	0	250 kW

Figure 6.6 shows the normal behavior of the load balance algorithm in the absence of a fake demand attack. Buses 2 and 3 both start with initial real power injections of 0 before performing a series of power migrations with each other. A power migration leads to coordinated power steps with identical magnitudes in opposite directions.

Bus 3 was then set to perform the fake demand attack and physical attestation was disabled. Figure 6.7 shows the impact of an undetected fake demand attack. Buses 2 and 3 form migration contracts as before, and Bus 2 ramps up its generation to fulfill its portion of the power migration. However, Bus 3 fails to perform its increase in load

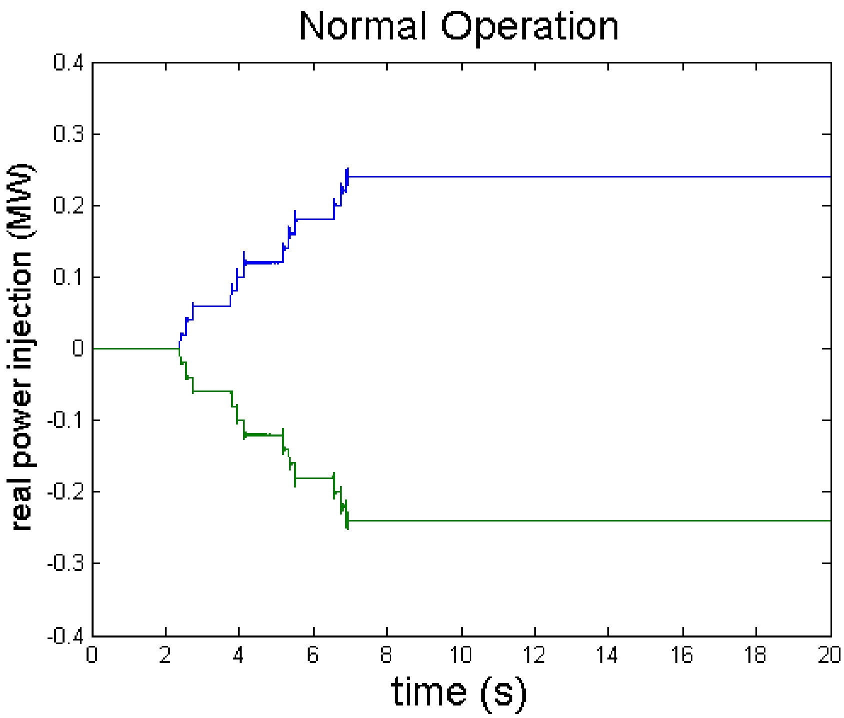


Figure 6.6: Normal Power Migration

and keeps its real power injection at a constant value of 0. This leads to excess generation being pushed into the distribution grid that must be handled by the isochronous generator $G1$.

Figure 6.8 shows the same case of the fake demand attack but performed when physical attestation is enabled. The round robin schedule of DGI causes short bursts of power migrations which step the power value up to the final amount of 285 kW. The other DGI algorithms are given time to run only after the time scheduled for load balance. Physical attestation executes during this load balance idle period and determines that a number of invalid power migrations have occurred. The attestation failure messages will not be received by load balance until it is scheduled to run again in its next iteration. This causes load balance to undo all the power migrations from the previous iteration, and Bus 2 resets its real power injection to 0. The jump from 285 kW to 0 demonstrates that physical attestation has worked, as the load balance algorithm has reset its power injection in response to notifications from attestation of bad power transactions.

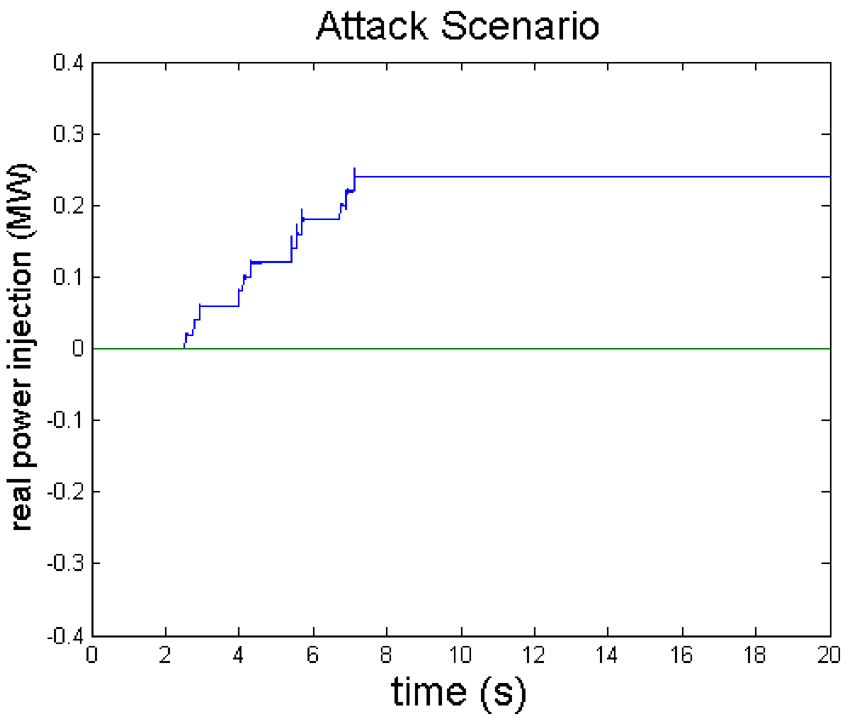


Figure 6.7: Fake Demand Attack

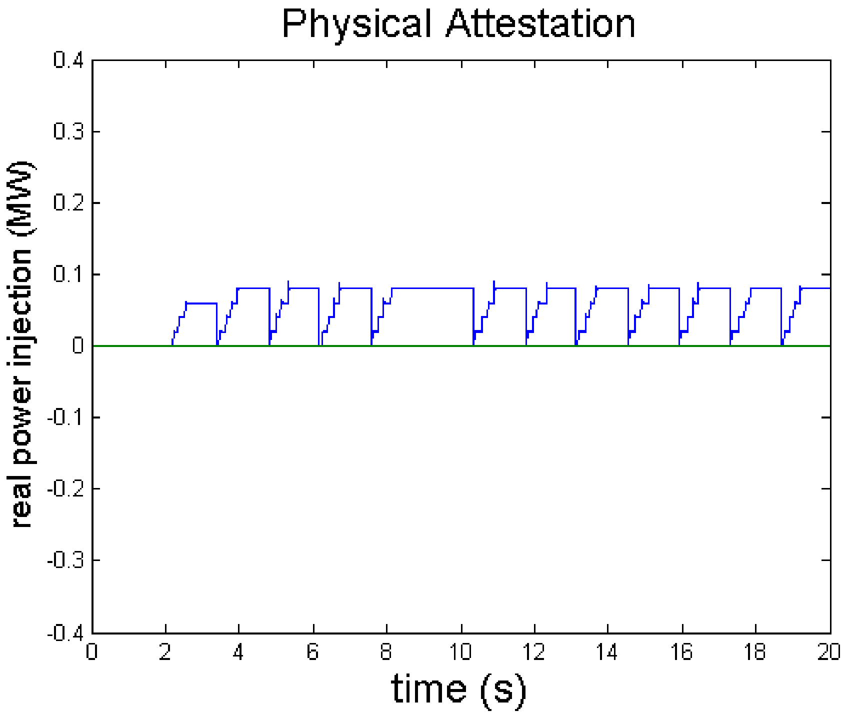


Figure 6.8: Physical Attestation

After Bus 2 resets its real power injection as a result of attestation failure, it still has an excess of 285 kW of transferable power. It therefore tries again to migrate its power to another entity in the system. However, the only load in the system with an attached DGI is the malicious Bus 3. As a result, Bus 2 always tries to migrate power with the malicious bus which causes the repeated behavior of power steps and resets in Figure 6.8. Load balance would have to be augmented with an additional security scheme such as a reputation system based on number of failed transactions in order to prevent this looping behavior.

7. CONCLUSION

This dissertation introduced a new distributed security mechanism called physical attestation for the smart grid. The theoretical foundation for physical attestation was proved for general graphs using graph theory and nondeducibility. A smart grid test bed was then utilized to produce experimental results that demonstrate the theory holds in practice. This section discusses the main contributions of this dissertation, and proposes several future research directions.

7.1. CONTRIBUTIONS

When physical attestation was proposed as a research topic, three goals were set for this dissertation. This section enumerates those goals, and describes the contributions this dissertation has made towards each of them. The following discussion describes how the three research goals listed below have been achieved.

1. Extend physical attestation to multiple cyber-physical system topologies
2. Develop an attestation framework applicable to multiple attack scenarios
3. Implement the theoretical results on a simulated cyber-physical system

7.1.1. Multiple System Topologies. The first research goal was to determine whether physical attestation could be applicable to multiple physical topologies. Theorem 2 enumerates the topological features that are both necessary and sufficient to apply physical attestation in general graphs. In order for this result to be useful in a distributed system with thousands of processes, Theorem 3 introduces the notion of an attestation framework that uses a subset of the system state. Finding the minimum size attestation framework for a general graph was proven to be NP-Hard in Theorem 5. Section 5 then went on to find polynomial-time solutions for ring, mesh, and hypercube topologies of arbitrary sizes.

7.1.2. General Attestation Framework. The second research goal was to formulate a general attestation implementation that works for multiple attack scenarios. Theorem 4 describes a polynomial time algorithm to generate attestation frameworks for general graphs that uses all vertices within four-hops of the attestation target. Algorithm 1 uses this framework to implement physical attestation in an energy management algorithm for the smart grid. The attack scenario considered in the experimental results was a fake demand attack in which a controller falsifies its broadcast load to trick suppliers into committing to bad power migrations. However, Lemma 1, Lemma 2, and Lemma 3 which form the backbone of this entire dissertation considered all possible false state attacks against the energy management algorithm. Therefore, Algorithm 1 can be used to protect the energy management algorithm against all possible false state attacks that involve a single compromised controller.

7.1.3. Experimental Implementation. The third research goal was to integrate physical attestation with an energy management algorithm and experiment on a simulated cyber-physical system. A smart grid test bed at the FREEDM Systems Center was utilized to perform the experiments. Figure 6.3 provides an overview of the physical attestation implementation. Figure 6.8 presents the main experimental result that demonstrates the feasibility of physical attestation. All of the code for the physical attestation implementation can be found on the DGI development github [45].

7.1.4. Refereed Publications. This dissertation reflects the following peer-reviewed publications:

1. 7th International Conference on Critical Information Infrastructures Security [46]
2. 8th International Conference on Critical Information Infrastructures Security [47]
3. IEEE Transactions on Dependable and Secure Computing [48]

[46] used nondeducibility to examine a series of false state attacks performed against energy management in the smart grid. Physical attestation as a security mechanism to detect and prevent these attacks was then introduced in [47]. The theoretical

proofs and experimental results that support physical attestation were then provided in [48].

7.2. FUTURE WORK

Physical attestation, even restricted to the single application of preventing false state attacks against energy management in the smart grid, is a large research area with many potential future research directions. Several of the deficient areas of this dissertation, as well as other exciting research directions, are enumerated in the following discussion.

7.2.1. Theoretical Extensions. The idea of physical attestation is to use physical feedback in a distributed cyber-physical system to detect malicious cyber behavior. This dissertation restricted itself to a single application that utilized the conservation of power in the smart grid to detect false state attacks against energy management. It is easy to imagine other applications of physical application that handle different forms of cyber misbehavior in different types of cyber-physical systems. With respect to different system types, the most immediate extension would be water distribution systems as the dynamics of water flow are similar to electric power flow. However, it should be possible to extend the main body of this dissertation to apply to most flow-based cyber-physical systems which move a commodity through some physical medium.

A more interesting research direction would be to consider how using different forms of physical feedback affects the deducibility of attacks. This dissertation chose the conservation of power as a physical invariant, and that choice determined the invariant violation patterns of the false state attack. However, other work in literature has considered a similar approach using Kirchhoff's voltage law to promising effect [49]. A different choice of physical invariant would change all of the invariant violation patterns and could lead to less nondeducible attack scenarios. One extension of this work would be to consider alternative invariants and the impact of invariant choice on attack deducibility.

7.2.2. Attestation Frameworks. The attestation framework presented in this dissertation can tolerate falsified states from a single compromised controller. Although the theory allows for multiple processes to be compromised so long as they participate in different attestation frameworks, physical attestation cannot tolerate collaborative attacks in the same framework. An easy extension of attestation frameworks would be to develop a framework that can tolerate multiple compromised controllers and still verify the attestation target.

The main limitation of attestation frameworks is that the problem of finding a minimum size framework in a general graph is NP-Hard. A significant future extension could develop approximation algorithms that generate good enough attestation frameworks for general graphs. Complexity analysis of these approximation algorithms in terms of both time complexity and framework size would be a worthwhile extension of this research.

7.2.3. Additional Experimentation. One unfortunate limitation during this dissertation was that the smart grid test bed was unable to support large power simulations. Experiments using a standard IEEE power system would greatly strengthen the physical attestation experimental results. Finding optimal frameworks in standard IEEE topologies could also generate a lot of insight into how attestation frameworks are formed.

One feature of the test bed that this dissertation was unable to explore was reconfiguration of both the physical topology and the communication network. Physical attestation requires measurements from a historic state of the system, and an interesting scenario would be attestation in the presence of either physical or network reconfigurations. There would also be significant merit in a more thorough integration of physical attestation with the DGI that enables attestation to be used by services other than load balance. The current implementation could be extended to be integrated with other distributed smart grid algorithms that utilize real power injection.

7.3. CONCLUDING REMARKS

The primary motivation of this research was the development of a distributed cyber security mechanism that utilizes physical feedback. Cyber-physical systems have the potential to cause drastic changes to existing cyber security mechanisms due to the existence of a physical layer. A research area that must be explored is how this physical layer both helps and hinders cyber security. This dissertation presented the idea of physical attestation in which cyber misbehavior is verified against the physical system. The main contribution of this work is that it serves as a case study that illustrates how physical feedback might be used to augment a distributed security mechanism.

BIBLIOGRAPHY

[1] Nicolas Falliere, Liam O Murchu, and Eric Chien. W32. Stuxnet dossier. *White paper, Symantec Corp., Security Response*, 2011.

[2] Adrian Perrig, John Stankovic, and David Wagner. Security in wireless sensor networks. *Communications of the ACM*, 47(6):53–57, 2004.

[3] Ravi Akella, Fanjun Meng, Derek Ditch, Bruce McMillin, and Mariesa Crow. Distributed power balancing for the FREEDM system. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 7–12. IEEE, 2010.

[4] Thomas J. Watson IBM Research Center. Research Division, BG Lindsay, PG Selinger, C Galtieri, JN Gray, RA Lorie, TG Price, F Putzolu, and BW Wade. *Notes on distributed databases*. 1979.

[5] Stephen McLaughlin, Dmitry Podkuiko, and Patrick McDaniel. Energy theft in the advanced metering infrastructure. In *Critical Information Infrastructures Security*, pages 176–187. Springer, 2010.

[6] Yao Liu, Peng Ning, and Michael K Reiter. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security (TISSEC)*, 14(1):13, 2011.

[7] Saman Zonouz, Katherine M Rogers, Robin Berthier, Rakesh B Bobba, William H Sanders, and Thomas J Overbye. SCPSE: Security-oriented cyber-physical state estimation for power grid critical infrastructures. *Smart Grid, IEEE Transactions on*, 3(4):1790–1799, 2012.

[8] Jie Lin, Wei Yu, Xinyu Yang, Guobin Xu, and Wei 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*, pages 183–192. IEEE, 2012.

[9] Tamal Paul, Jonathan W Kimball, Maciej Zawodniok, Thomas P Roth, Bruce McMillin, and Sriram Chellappan. Unified invariants for cyber-physical switched system stability. *Smart Grid, IEEE Transactions on*, 5(1):112–120, 2014.

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

[11] Gerry Howser and Bruce McMillin. A multiple security domain model of a drive-by-wire system. In *Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual*, pages 369–374. IEEE, 2013.

[12] Steve H. Weingart. Physical security devices for computer subsystems: A survey of attacks and defenses. In *Cryptographic Hardware and Embedded Systems ECHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 302–317. Springer Berlin Heidelberg, 2000.

[13] G. Edward Suh, Dwaine Clarke, Blaise Gassend, Marten van Dijk, and Srinivas Devadas. AEGIS: architecture for tamper-evident and tamper-resistant processing. In *Proceedings of the 17th annual international conference on Supercomputing*, ICS '03, pages 160–171, New York, NY, USA, 2003. ACM.

[14] Ross J. Anderson and Markus G. Kuhn. Low cost attacks on tamper resistant devices. In *Proceedings of the 5th International Workshop on Security Protocols*, pages 125–136, London, UK, UK, 1998. Springer-Verlag.

[15] Ross Anderson and Roger Needham. Programming satan's computer. In Jan Leeuwen, editor, *Computer Science Today*, volume 1000 of *Lecture Notes in Computer Science*, pages 426–440. Springer Berlin Heidelberg, 1995.

[16] Bill Horne, Lesley Matheson, Casey Sheehan, and Robert E. Tarjan. Dynamic self-checking techniques for improved tamper resistance. In Tomas Sander, editor, *Security and Privacy in Digital Rights Management*, volume 2320 of *Lecture Notes in Computer Science*, pages 141–159. Springer Berlin Heidelberg, 2002.

[17] Hoi Chang and MikhailJ. Atallah. Protecting software code by guards. In Tomas Sander, editor, *Security and Privacy in Digital Rights Management*, volume 2320 of *Lecture Notes in Computer Science*, pages 160–175. Springer Berlin Heidelberg, 2002.

[18] Fred C Schweppe and Douglas B Rom. Power system static-state estimation, part ii: Approximate model. *power apparatus and systems, ieee transactions on*, (1):125–130, 1970.

[19] E Handschin, FC Schweppe, Ji Kohlas, and AAFA Fiechter. Bad data analysis for power system state estimation. *Power Apparatus and Systems, IEEE Transactions on*, 94(2):329–337, 1975.

[20] Yao Liu, Peng Ning, and Michael K. Reiter. False data injection attacks against state estimation in electric power grids. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 21–32, New York, NY, USA, 2009. ACM.

[21] Jinsub Kim and Lang Tong. On topology attack of a smart grid: Undetectable attacks and countermeasures. *Selected Areas in Communications, IEEE Journal on*, 31(7):1294–1305, 2013.

[22] Oliver Kosut, Liyan Jia, Robert J Thomas, and Lang Tong. Malicious data attacks on the smart grid. *Smart Grid, IEEE Transactions on*, 2(4):645–658, 2011.

[23] Annarita Giani, Eilyan Bitar, Manuel Garcia, Miles McQueen, Pramod Khargonekar, Kameshwar Poolla, et al. Smart grid data integrity attacks. *Smart Grid, IEEE Transactions on*, 4(3):1244–1253, 2013.

[24] Jian Chen and Ali Abur. Placement of pmus to enable bad data detection in state estimation. *Power Systems, IEEE Transactions on*, 21(4):1608–1615, 2006.

[25] György Dán and Henrik Sandberg. Stealth attacks and protection schemes for state estimators in power systems. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 214–219. IEEE, 2010.

[26] Henrik Sandberg, André Teixeira, and Karl H Johansson. On security indices for state estimators in power networks. In *First Workshop on Secure Control Systems (SCS), Stockholm, 2010*, 2010.

[27] Rakesh B Bobba, Katherine M Rogers, Qiyan Wang, Himanshu Khurana, Klara Nahrstedt, and Thomas J Overbye. Detecting false data injection attacks on dc state estimation. In *Preprints of the First Workshop on Secure Control Systems, CPSWEEK*, volume 2010, 2010.

[28] Ali Tajer, Soummyar Kar, H Vincent Poor, and Shuguang Cui. Distributed joint cyber attack detection and state recovery in smart grids. In *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*, pages 202–207. IEEE, 2011.

[29] Kin Cheong Sou, Henrik Sandberg, and Karl H Johansson. Data attack isolation in power networks using secure voltage magnitude measurements. *Smart Grid, IEEE Transactions on*, 5(1):14–28, 2014.

[30] D.E. Denning. An intrusion-detection model. *Software Engineering, IEEE Transactions on*, SE-13(2):222–232, 1987.

[31] H.S. Javitz and A. Valdes. The SRI IDES statistical anomaly detector. In *Research in Security and Privacy, 1991. Proceedings., 1991 IEEE Computer Society Symposium on*, pages 316–326, 1991.

[32] Jake Ryan, Meng jang Lin, and Risto Miikkulainen. Intrusion detection with neural networks. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, pages 943–949. MIT Press, 1998.

[33] K. Ilgun, R.A. Kemmerer, and P.A. Porras. State transition analysis: a rule-based intrusion detection approach. *Software Engineering, IEEE Transactions on*, 21(3):181–199, 1995.

[34] Franco P. Preparata, G. Metze, and Robert T. Chien. On the connection assignment problem of diagnosable systems. *Electronic Computers, IEEE Transactions on*, EC-16(6):848–854, 1967.

[35] S. Mallela and G.M. Masson. Diagnosable systems for intermittent faults. *Computers, IEEE Transactions on*, C-27(6):560–566, 1978.

[36] S. Mallela and G.M. Masson. Diagnosis without repair for hybrid fault situations. *Computers, IEEE Transactions on*, C-29(6):461–470, 1980.

[37] A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla. Swatt: software-based attestation for embedded devices. In *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*, pages 272–282, 2004.

[38] Kyungsub Song, Dongwon Seo, Haemin Park, Heejo Lee, and A. Perrig. Omap: One-way memory attestation protocol for smart meters. In *Parallel and Distributed Processing with Applications Workshops (ISPAW), 2011 Ninth IEEE International Symposium on*, pages 111–118, 2011.

[39] Haemin Park, Dongwon Seo, Heejo Lee, and Adrian Perrig. Smatt: Smart meter attestation using multiple target selection and copy-proof memory. In Sang-Soo Yeo, Yi Pan, Yang Sun Lee, and Hang Bae Chang, editors, *Computer Science and its Applications*, volume 203 of *Lecture Notes in Electrical Engineering*, pages 875–887. Springer Netherlands, 2012.

[40] Yi Yang, Xinran Wang, Sencun Zhu, and Guohong Cao. Distributed software-based attestation for node compromise detection in sensor networks. In *Reliable Distributed Systems, 2007. SRDS 2007. 26th IEEE International Symposium on*, pages 219–230, 2007.

[41] Thoshitha T Gamage, Thomas P Roth, and Bruce M McMillin. Confidentiality preserving security properties for cyber-physical systems. In *Computer Software and Applications Conference (COMPSAC), 2011 IEEE 35th Annual*, pages 28–37. IEEE, 2011.

[42] Alex Q Huang, Mariesa L Crow, Gerald Thomas Heydt, Jim P Zheng, and Steiner J Dale. The future renewable electric energy delivery and management (freedm) system: the energy internet. *Proceedings of the IEEE*, 99(1):133–148, 2011.

[43] Fanjun Meng, Ravi Akella, Mariesa L Crow, and Bruce McMillin. Distributed grid intelligence for future microgrid with renewable sources and storage. In *North American Power Symposium (NAPS), 2010*, pages 1–6. IEEE, 2010.

[44] The center for advanced power systems (CAPS). Florida State University, Web. 29 September 2015 (http://www.caps.fsu.edu).

[45] Distributed grid intelligence. Branch Attestation. Missouri University of Science and Technology, Web. October 2 2015 (https://github.com/FREEDM-DGI/FREEDM).

[46] Thomas Roth and Bruce McMillin. Breaking nondeducible attacks on the smart grid. In *Seventh CRITIS Conference on Critical Information Infrastructures Security*, Lillehammer, Norway, 2012. Springer.

[47] Thomas Roth and Bruce McMillin. Physical attestation of cyber processes in the smart grid. In *Eighth CRITIS Conference on Critical Information Infrastructures Security*, Amsterdam, Netherlands, 2013. Springer.

[48] Thomas Roth and Bruce McMillin. Physical attestation in the smart grid for distributed state verification. *Dependable and Secure Computing, IEEE Transactions on.* Under Review.

[49] Iman Shames, André MH Teixeira, Henrik Sandberg, and Karl H Johansson. Fault detection and mitigation in kirchhoff networks. *Signal Processing Letters, IEEE*, 19(11):749–752, 2012.

VITA

Thomas Patrick Roth was born in St. Louis, Missouri. He received his Bachelor's degree in both Computer Science and Computer Engineering from Missouri University of Science and Technology in May, 2011. He then joined the Computer Science Ph.D. program at the same university in the fall of 2011 with Dr. Bruce McMillin as his research advisor. He received his Ph.D. in December 2015 and graduated summa cum laude. After graduation, he went on to work at the National Institute of Standards and Technology as part of its Cyber-Physical Systems and Smart Grid program. His research interests are in information flow security and the detection of compromised processes in distributed cyber-physical systems.