Tamper Event Detection on Distributed Devices in Critical Infrastructure

Jason Reeves, Sean Smith
Dartmouth College
{reeves, sws}@cs.dartmouth.edu

Abstract

Utilities are currently installing a number of resource-constrained embedded devices at the remote endpoints of their SCADA (Supervisory Control and Data Acquisition) networks as part of their smart grid rollout. These devices present a security risk for utilities: They are deployed in many different places and environments; they have very little physical security; and they have direct access back to control centers and other infrastructure. Therefore, these devices present an easy opportunity for attackers to infiltrate and damage a utility's SCADA network, and protecting these networks is critical.

While a large body of work exists in both the physical tampering and SCADA protection spheres, current schemes cannot be applied here: They are either not designed to protect both a device and the network it lives on, not flexible enough to handle the various tamper events that these remote devices face, or not lightweight enough to operate under the inherent constraints of a SCADA network. In this paper, we introduce T.E.D.D.I. (Tamper Event Detection on Distributed Infrastructure), a distributed, sensor-based tamper protection architecture that we are building to protect remotely-deployed devices. This architecture will allow for a flexible, policy-centric response to tamper events, incorporate external context data into T.E.D.D.I.'s decision-making process, and adhere to the many constraints imposed on SCADA systems. We also discuss the current state of T.E.D.D.I., its various components, and our plans for constructing and validating the final system.

1 Introduction

As a real-time system, today's power grid requires constant monitoring and timely responses to maintain proper operations, often within time frames beyond which a human could react. The speed required for this kind of control has led the industry to automate many routine tasks based on the state of the grid, such as generator governors and protective relays [29].

More recently, utilities have introduced a number of smart technologies to improve the grid's reliability and efficiency. This push has introduced a small number of resource-constrained embedded devices into the grid, including at the remote endpoints of a utility's SCADA[1] network. One example of such a device is a recloser control [19]. Often mounted inside boxes on utility poles in the field, recloser controls are used to configure how a utility's reclosers behave when a fault is detected in the power lines.

Deploying this kind of technology presents a major security challenge for three reasons: These devices are easily accessible, they have very little physical security, and they have direct access to a utility's SCADA network. Therefore, remotely-deployed smart devices provide an easily-accessible gateway to a utility's broader communication and control network, making them an enticing target for attackers looking to damage the grid. Protecting these devices thus becomes a high priority for a utility.

While a large corpus of work exists in the area of tamper detection and physical security, this work is not sufficient to protect remotely-deployed grid devices for a number of reasons:

- Often tamper and intrusion solutions focus solely on the device or the network they live on, whereas here we need to protect devices *and* the larger SCADA network.
- Many tamper solutions are unable to differentiate between important tamper events. For example, if the device senses that it is being accessed, is it by a malicious attacker, a utility service technician, or something different altogether, such as a natural disaster?
- Current SCADA protection solutions (for example, [25] and [30]) generally just report a problem when something happens, and make no attempt to respond.
- Tamper solutions in the FIPS 140-2 sphere (for example, [10], [23], and [32]) only have a single response to problems (usualy destroying data and rendering the device unusable), and cannot adjust their strategy based on the type of tampering detected. This is a major conern for the power grid, since taking an unnecessarily-severe action may reduce the grid's availability.
- Some detection systems cannot operate under the inherent constraints of a SCADA network. This

[1] SCADA stands for "Supervisory Control and Data Acquisition," and is generally used to describe the command-and-control networks used by critical industries such as water, gas, and electricity.

problem extends to both network constraints (the systems may interfere with important traffic and delay it too long to be useful) and geographic constraints (SCADA devices operate in a wider range of environments than current tamper solutions).

To address these issues, we propose T.E.D.D.I (Tamper Event Detection on Distributed Infrastructure), a distributed, sensor-based system that can use information from the larger network to help inform tamper decisions at the local level.

Our goal is to construct a tool that SCADA operators can use to generate the necessary protection components for an arbitrary network topology, sensor set, and group of concerning events. Currently, we have a small prototype of our system, and will use it to facilitate the construction of our system-generating tool. We also plan to include a *program placement tool* to help operators decide where to place system components within their network, and an *event estimation tool* to allow administrators to see how small changes in event probabilities might affect the overall detection algorithm.

The contributions of this work are as follows:

- We will build a novel system that addresses the security concerns surrounding endpoint devices in a SCADA network and protects both these devices and the network from being tampered with by malicious actors. The system will be specifically geared towards protecting remotely-deployed embedded devices in the power grid, and will adhere to the grid's resource and environmental constraints. We will also build a prototype of our system and evaluate its performance both in a simulated environment and in a testbed on actual power hardware.
- Our system will allow for a range of responses to a tamper event. This means that we can adjust our response to a tamper event based on the system configuration and the decision it makes, tune the system to a specific point on the availability-vs.-security spectrum, and account for "legitimate" tamper events, e.g., technician service visits.
- Our system will also feature a novel way to make tamper decisions based on the information the system has available. At the local level, components embedded with remote devices will fuse sensor data together using *factor graphs* [8] to figure out what event is occurring, while also reporting its current

state to a *decision point* deeper within the network. In cases where the local device cannot discern its exact state, it will ask the decision point to make a determination based on its larger information base, and use that larger data set to choose the proper responses.

The rest of this paper is structured as follows: Section 2 discusses important background information, including an explanation of factor graphs, Section 3 introduces the relevant related work in this space, Section 4 introduces T.E.D.D.I. and explains its individual components, Section 5 discusses T.E.D.D.I.'s current status and lays out our future plans, and Section 6 contains a set of conclusions about our recent work.

2 Background

In this section, we will discuss the attack vectors created by remote SCADA devices, the problems with fitting an existing tamper protection scheme to a SCADA network, and the benefits of choosing factor graphs as our data fusion algorithm.

2.1 The Problem with Remote SCADA Devices

Remote SCADA devices pose a large security risk to a utility for three reasons:

1. Devices such as recloser controls are distributed all across a utility's service area and may appear in almost any environment, from remote rural areas (where they are under very little supervision) to highly-populated urban areas (where they are easily accessible to a large number of people).
2. These devices often have little in the way of physical security, with nothing in between themselves and the outside world but the cabinet they are placed in.
3. These devices have a direct connection to a utility's SCADA network to communicate back to a utility's control center, which may also grant access to other devices on the network, other control centers, or perhaps other pieces of the control infrastructure.

In short, remote SCADA devices are a soft target for attackers to compromise for their own nefarious purposes. Once an attacker gains access to the device, two avenues are open to them:

- They could compromise the SCADA device itself, and use it to send bad data back to the utility or interfere with the communication of other SCADA devices.
- They could bypass the device altogether, and instead plug their own hardware into the device's access point and use it as a launching point for further attacks on high-value targets deeper in a utility's network.

The latter scenario is what worries utility operators the most: A recent *Wall Street Journal* article [21] declared that malicious attackers could cause a nationwide blackout by taking down less than ten critical substations during a period of high-demand on the grid, and that such a blackout "could plunge the country into darkness for weeks, if not months" [21]. A compromised remote device could give an attacker access to one or more of these critical substations, and thereby allow them to cause damage that extends far beyond the loss of a single device.

2.2 Tamper Detection and the Power Grid

As we have discussed in prior work [17], a number of *intelligent electronic devices* have been introduced in the push to create a smarter electric grid. While the capabilities of these devices can vary widely, they are generally weaker than general-purpose computing systems—for example, Motorola's ACE3600 Remote Terminal Unit only boasts a 200 mHz microprocessor, 16 MB of Flash memory, and 32 MB of DRAM [2]. In addition, the grid's SCADA network itself imposes strict network and timing constraints, as they must send data across "a partially unsecured slow legacy network" [24] within some very small timing windows [1]. Causing these devices to miss their timing windows keeps them from doing their job properly and therefore is just as problematic as having a compromised device.

Other issues include:

- The attack surface we need to protect. Our primary goal is to keep attackers from accessing a SCADA

network via one of these remote devices. An attacker could gain this access in two ways (see Section 2.1), so we need to guard against both possibilities.[2]

- The various kinds of "tampering" these devices face. While malicious attackers are our biggest concern, these devices are also accessed by service technicians, and may be impacted by environmental events such as weather or natural disasters. In each case, a different response may be required, since the grid's goal is to maximize availability while still protecting its network from attack.

These issues have made designing a tamper protection system suitable for the power grid and other SCADA networks difficult, as any such solution has to cover a broad attack surface and handle several different types of tampering while not getting in the way of any device's primary functions.

2.3 Data Fusion

A big question for any tamper protection system is how it fuses the data it collects to make a decision. This fusion is usually done using a standard technique such as Bayesian networks [16], a custom algorithm such as SCADAHawk's snapshots [25] or Amilyzer's flow matching [4], or a combination of the two (for example, RRE's attack response trees, Markov decision processes, and Stackelberg games [37]). In general, however, these fusion methods are tuned to certain types of attack models at the expense of others. Bayesian analyses, for example, are geared towards looking for the simple presence or absence of indicators, and adapting them to other possibilities, such as specific sequences of indicators, can make these models infeasibly complex. Flow matching, on the other hand, looks for specific sequences or identifiers, and might miss slight deviations from these paths that are still malicious.

Instead, we plan to leverage the additional power of *factor graphs* [8]. Formally, a factor graph is a bipartite graph that connects a set of nodes V that represents system variables with a set of nodes F that represents functions relating these variables. If a function is dependent on a variable, an edge is added to the graph between the nodes that represent this variable and function.

On the surface, factor graphs feature a setup similar to a Bayesian network, complete with nodes that

[2]Note that we do not consider an attacker remotely compromising one of these devices as within the scope of our problem.

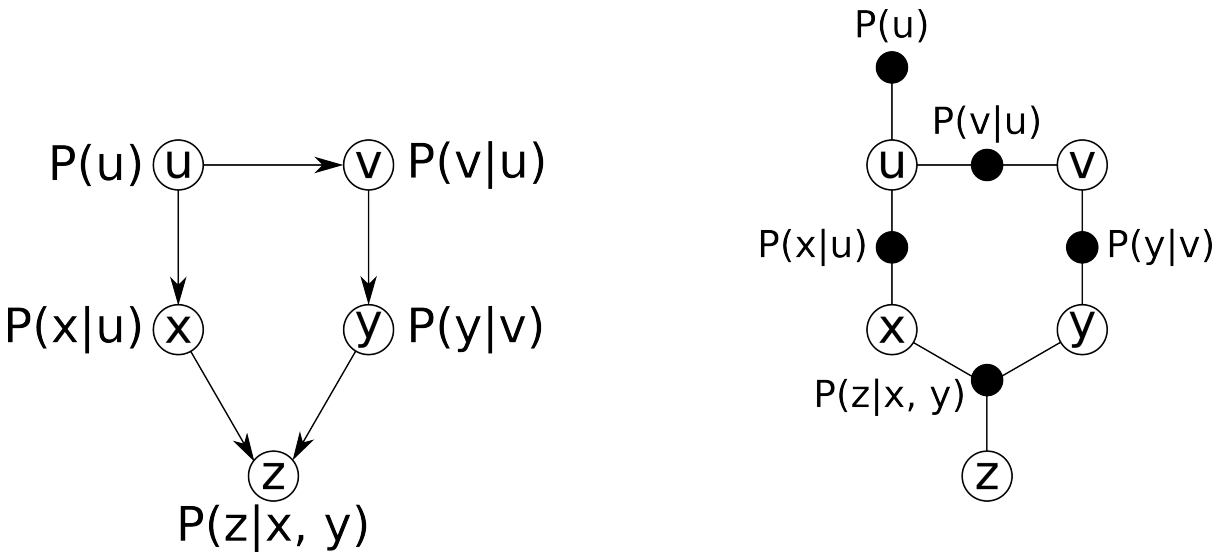

Figure 1: Figures 2a (left) and 2b (right) from [8], depicting the differences between a Bayesian network and factor graph that solve the exact same problem. Note that the Bayesian network (left) must be directed, but the factor graph can be either directed or undirected.

represent indicators and edges that denote the dependencies between indicators. (See Figure 1 for a comparison of a Bayesian network and equivalent factor graph.) However, instead on relying solely on conditional probabilities, factor graphs allow for arbitrary functions to be used to describe indicator dependencies. This extra power allows us to overlay additional techniques on top of conditional probabilities to increase the attack coverage of our system without adding unnecessary complexity to the model.

For example, consider a scenario in which a *specific sequence of indicators* signals the presence of a completely different event than what those indicators might signal by themselves. If this sequence characterizes an attack, an attacker can easily evade a Bayesian-based system by making sure the indicators its sequence generates never appear together, since a Bayesian network only considers the presence or absence of indicators. However, with factor graphs we can incorporate things such as indicator histories and inter-indicator relationships into our detection algorithm, and thus catch the attacker's suspicious sequence. In addition, we can also specify a response sequence to coordinate our actions and further improve our defensive stance. Thus, *using factor graphs gives us improved event detection and protection capabilities over current SCADA protection systems.*

Factor graphs have been applied to a number of different fields in the past, including signal processing [14], robot navigation [11], and evaluating trust on ad-hoc networks [26]. To the best of our knowledge, however, this is the first time factor graphs have been applied to the protection of SCADA devices from tampering. See Section 4.1 for more information about the specific factor graph in T.E.D.D.I.

3 Related Work

In this section, we will discuss some of the related work in both hardware tamper protection and SCADA network protection, and show why both fields fall short in addressing the issue of remote SCADA devices.

3.1 Hardware Tamper Protection

Hardware tamper protections are what most people think about when they consider classic tamper solutions. The roots of this area of research run deep, and include seminal works such as Kent's ideas for tamper-resistant modules [13], White and Comerford's ABYSS platform [34], White et al.'s work on the Citadel system [35], Tygar and Yee's Dyad platform [27], and Smith, Palmer, and Weingart's work on the IBM 4758 [22, 23]. Today, a number of hardware solutions are available commercially, either in the form of trusted platform modules (for example, [3, 9, 12]) or cryptographic coprocessors [10]. However, these solutions are geared towards single-device protection, and may have trouble operating in the extreme conditions power devices must face. For example, the IBM 4765 is only designed to run within a temperature range of 10 to 35 degrees Centigrade [10], while the SEL-651R recloser control must be able to run within a temperature range of -40 to 55 degrees [20].

In recent academic work, Dragone [6] uses several layers of *patches* connected by a wire mesh and laid out in a random pattern such that anyone attempting to drill through the material could not avoid hitting a patch and triggering a response, while Megalingam et al. [15] discusses connecting a smart meter's power supply to the screws that hold its case together, rendering the meter useless if anyone tries to open the case and access its internals. Desai [5], on the other hand, takes an obfuscation approach by adding extra dummy states to a chip's finite state machine, and only transitioning to the true functional states if a special code-

word is provided to the chip. Once again, however, these solutions take too limited a view of the problem, and do not offer protection against the bypass attack vector from Section 2.1.

3.2 SCADA Network Protection

Berthier and Sanders's Amilyzer [4] is a specification-based network intrusion detection system developed for the advanced metering infrastructure being rolled out in the smart grid. Amilyzer monitors traffic from AMI devices and organizes the packets into *flows*, which are then scrutinized for policy violations or other unwanted behaviors. However, Amilyzer does not consider physical tampering in its current form, and is still looking into the idea of being "distributed and coordinated across multiple sensors" [4].

Roblee, Berk, and Cybenko's Process Query Systems [18] combine host and network monitoring to determine which nodes in a network might be compromised. Process sensors at the host level report information back to PQS's fusion engine, which uses conditional probabilities to relate events to one of its attack/failure models. While the system focuses on bad behavior within devices and their network, the system could potentially be configured to handle environmental events as well. However, trying to track a large number of behavior models can be costly, and leaves the system open to unknown attacks that do not match existing models.

Sousan et al.'s SCADAHawk system [25] uses a system of *collectors* (low-level signal monitors) and *agents* (storage programs for collector data) to learn what behaviors are considered normal in different parts of the network. Data from the collectors is categorized according to the system's event taxonomy and organized by agents into event sequences called *snapshots* that capture normal behavior on the system. The agents can then be shifted into a monitoring mode that looks for behavior that deviates from the snapshots. SCADAHawk is a detection system only, however, and having to compare an observation to every single snapshot in the system will be costly.

Valdes and Skinner's Probabilistic Event Correlation system [28] uses Bayesian-based data fusion as a way to reduce false positives within an intrusion detection system. The system maintains a list of *meta alerts* that might represent an attack, and adds individual alerts to the meta alert if the system thinks they are

similar. The system also maintains a minimum similarity threshold and a priority field within the meta alert, all with the goal of showing the administrator only the issues that are most likely to be security violations. The similarity metric, however, can become cumbersome as the number of meta alerts and alert features grow, and attack class similarity is evaluated via a static matrix that will require updating as different threats emerge.

Wang and Hauser's Bayesian trust assessment framework [30] tries to evaluate the trust they can place in a device based on the evidence they collect. The authors collect a series of data vectors within a small time window, define a loss function that captures the consequences of taking an action a when the device's trust level is t, and finally use a parameterized risk function to decide whether or not the device is trustworthy or not. However, the program suffers from the weakness of its Bayesian basis, and only makes a binary trust decision that is not granular enough to extend to intrusion response. Additionally, its only inclusion of user preferences is in its risk function, where it allows the user to say whether they are more risk-tolerant or risk-averse.

Zonouz et al.'s SCPSE [38] is a fusion framework that combines intrusion detection alerts with power system information to more accurately estimate the security state of an electrical network. SCPSE builds an attack graph that traces the possible paths an attacker could follow via exploiting network nodes, and determines how power information in the system should be correlated to devices in the network. In its monitoring mode, SCPSE uses power flow information and intrusion alerts to estimate the attacker's path through the graph, and thus reveal which devices in the network are potentially compromised. Despite its increased awareness, however, this system still ignores external environmental factors that might affect the network's security state, and does not incorporate possible responses into its design.

The system that most closely resembles ours is Zonouz et al.'s Response and Recovery Engine [37]. The RRE uses *attack response trees* to define the security goals it wants to maintain, the various ways these goals might be violated, and the possible responses that could be taken to maintain those goals. As intrusion alerts come in, the system determines which nodes in the tree have been reached, which represents what the attacker has achieved thus far. The trees are converted to Markov decision processes, which are then solved to determine the optimal action to take against the attacker. The RRE also has local and global components,

which allows it to monitor both the state of individual boxes and the overall state of the network. However, the RRE currently does not consider external events such as environmental factors, and the response trees have to be complete enough to cover the entire attack space that the administrator is concerned about.

4 T.E.D.D.I. Overview

In this section, we will discuss the architecture of the T.E.D.D.I. system, covering both the system components and the tool that generates them.

4.1 T.E.D.D.I. Factor Graphs

At the heart of our tamper decision engine is a factor graph [8] that calculates the probability that an event is currently occurring to a local device. These probabilities are based on two data sources:

Incident Data. The initial probabilities of events occurring will be grounded in incident data collected by the utilities—for example, how often a device sees its cabinet opened, or how often it is shaken by an earthquake. Given the vast amount of data required, however, most utilities will have an incomplete set for their system.

Domain Expert Preferences. To fill the gaps in an incident data set, and to allow a utility to tweak their system's sensitivity towards certain events, we will also allow domain experts to set the probabilities in the system. We describe some of the tools we will provide for these experts in Section 4.3.

Our graph is constructed from three important datasets:

Events: The set $E = \{e_1, \ldots, e_j\}$ of events we want to detect.

Indicators: The set $I = \{i_1, \ldots, i_k\}$ of phenomena connected to the events in E. For example, if E includes an earthquake in its set, I might include shaking.

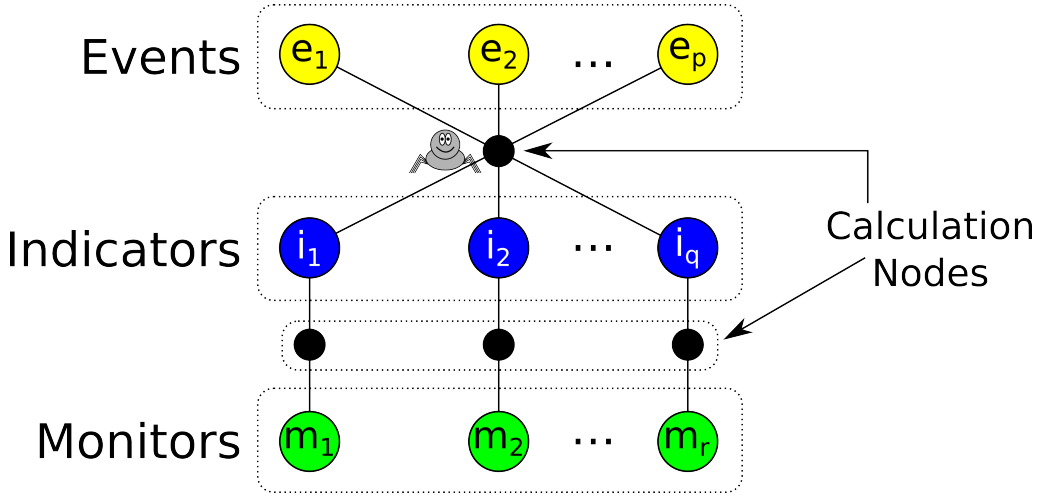

Figure 2: A complete factor graph generated by T.E.D.D.I. In general, each indicator is joined with a single monitor, while all of the events and indicators are connected through the spider node. However, the number of indicators and monitors does not necessarily have to be equal.

Monitors: The set $M = \{m_1, \ldots, m_l\}$ of sensors we are using to look for indicators. For our previous example, M might include an accelerometer or seismograph to look for intense shaking.[3]

We build the factor graph by defining the set of variable nodes V as the elements of E, I, and M, and building the set of function nodes F and the set of graph edges in the following manner:

- If a monitor m_p detects an indicator i_q, an intermediate calculation node c is added to F, and the edges $\{m_p, c\}$ and $\{c, i_q\}$ are added to the graph.
- A calculation node s is added to F, as well as the edges $\{e_p, s\}$ and $\{s, i_q\}$, $1 \leq p \leq j, 1 \leq q \leq k$. We refer to s as the *spider node*, and explain its purpose below.

Figure 2 shows an example of a generated graph.

The presence of the spider node gives our detection engine additional power beyond a simple Bayesian network, and allows us to make better, more accurate decisions about what is happening to a device. Because factor graphs allow for arbitrary functions to relate nodes to each other, we can use this node to look for

[3] In most cases, this means the number of indicators and monitors will be the same, i.e. $k = l$. However, this does not have to be true.

complex event/indicator relationships beyond conditional probabilities. A good example of this is *sequence tracking*: By routing our event/indicator paths through the spider node, we can look for inter-indicator relationships and spot sequences that indicate an attacker is in the process of compromising a device.

We actually generate two versions of the factor graph: A full version for tamper components with full information access, and a *limited* version for components that can only see a subset of the monitors and indicators. The limited graph differs from the full version in two ways:

- If a monitor is not accessible from a remote device, the monitor and corresponding indicator are removed from the limited factor graph.
- If two events are not distinguishable from a remote device, those events are combined into a single node in the limited factor graph.

4.2 T.E.D.D.I. Components

At a high level, our protection system will feature three major components: tamper information points, tamper decision points, and tamper enforcement points.[4] We now describe each component in detail.

4.2.1 Tamper Information Points (TIPs)

Tamper information points are the eyes and ears of the system, and are responsible for the detection and reporting of potential tamper events affecting the devices they protect. A TIP is assigned to each remote device in the network, and lives in the same location as that device (for example, in the same cabinet) monitoring the surrounding environment. Every few seconds, the TIP takes a snapshot of its sensor values and analyzes this snapshot to see what events, if any, are presently affecting the device. This data is sent to a tamper decision point as a matter of course, both to keep the decision point's worldview current and to alert the decision point that this TIP is still active.

As part of its data analysis, the TIP maintains threshold values for every monitor that it has. When a

[4] These names are inspired by the policy information, decision, and enforcement points from XACML [7, 33].

sensor value exceeds its corresponding threshold, the monitoring node is considered *on* for the purposes of our analysis, and *off* otherwise. We then use the graph to calculate $P(e_p|m_1, \ldots, m_t)$, $1 \leq p \leq j$ to see which event (if any) is occurring at the moment.

Most of the time, the TIP will be able to make its own decisions about what is happening using its copy of the factor graph. However, the TIP only receives the limited copy of the graph, since it will not have full access to the monitor and indicator set. When the TIP encounters a situation when it cannot differentiate between important events, it sends a request to a tamper decision point for assistance.

4.2.2 Tamper Decision Points (TDPs)

Tamper decision points live in higher-security areas of a utility's SCADA network, such as within a substation, and they serve multiple TIPs within their service area. TDPs have a full copy of the system's factor graph and complete access to the system's monitors, and thus they can make a more-informed tamper decision than the TIPs it serves.

When a TDP receives an assistance request from one of its TIPs, the TDP first tries to determine the overall state of the area it monitors. It does this by combining the heartbeat data it receives from all of its TIPs to see what sensor values are held by the majority of its TIPs. More specifically, if a TDP serves p TIPs, and $M_q = \{m_1, \ldots, m_l\} =$ the current sensor values of TIP q, then the TDP's state $M_{overall}$ is set as follows:

$$M_{overall} = \{\lfloor((\sum_p^1 m_1)/p) + .5\rfloor, \lfloor((\sum_p^1 m_2)/p) + .5\rfloor, \ldots, \lfloor((\sum_p^1 m_l)/p) + .5\rfloor\}$$

If a TIP stops reporting to the TDP, its data is eventually declared stale, and the offending TIP is removed from the summation.

Once the TDP calculates its overall state, it runs that state through the full factor graph to make a final decision as to what event is occurring. If the decided event exceeds our minimum event threshold, then that event as passed along to the appropriate tamper enforcement points to initiate the proper response.

Finally, as our tamper decision points collect state information and make decisions about local events, they will also pass data to a centralized aggregator that a utility can use to see the tamper states of all the

devices in their network. The aggregator will format this data for use within a network visualization tool such as CPTL [31].

4.2.3 Tamper Enforcement Points (TEPs)

Tamper enforcement points live in between a TDP and TIP, and they are responsible for responding to decisions made by these devices. A TEP is associated with a single device and TIP, and has a set R of responses that it can execute when needed.

The TEP is not limited to making a single response to a decision, but can instead execute an ordered series of responses to mitigate any problems. It considers the following when deciding on an optimal response:

Applicability: Does taking this response make sense in the context of the given event? For example, cutting off network access to a box in the middle of a hurricane does not make sense, given the grid's focus on resilience and availability.

Mitigation Amount: How effective is the response at mitigating the damage from the event? For example, destroying sensitive data on a remote device keeps the data out of the attacker's hands, but still leaves the network open to compromise.

Cost: How expensive in terms of time, money, damages, etc. is the event if left unchecked, and how costly is the response we are considering? These costs can be hard to quantify, and thus we are reliant on the utility to define them.

Timing: If a response is chosen, when should it happen? This is mostly a concern for when the TEP decides to execute multiple responses, since the order in which these responses are taken could affect their usefulness.

Once these factors are taken into account, the TEP can evaluate the benefits and drawbacks of each response, and come up with an appropriate set $R' \subset R$ of responses to take to protect the SCADA network.

4.3 T.E.D.D.I. Program Generation Tool

As currently constructed, T.E.D.D.I. requires a lot of customization to fit the network of a given utility, since it must be configured to the available sensors, devices, and concerning events. Therefore, our plan is to construct a tool that utility operators can use to generate the proper TIP, TDP, and TEP programs for any arbitrary network. While the interface is not finalized at this time, it will include:

- An interface for users to upload a network topology file for the system to analyze and determine a best-possible TIP/TDP/TEP layout for that given network. This layout would be shown to the user, who could then tweak the layout based on their monitoring preferences.
- An interface for entering incident data into the system, as well as a pre-existing database based on data that we collect (for example, statistics from partner utilities and government organizations) to reduce the amount of data needed from the utility.
- An *event estimation tool* that will allow users to see how changes they make to individual probabilities affect the overall event detection system. This will let users see how volatile their numbers are—for example, do small changes to one probability lead to big changes in our overall event probabilities?—and tweak their preferences in favor of false positives or false negatives depending on the events.

4.4 How T.E.D.D.I. Works: An Example

To clarify how the pieces of T.E.D.D.I. fit together, consider the simple example illustrated in Figure 3:

1. A utility operator uses the generation tool to construct a simple tamper system consisting of a TDP managing three TIPs A, B, and C. The operator denotes in the tool that the TIPs are equipped with an accelerometer as part of their sensor set, and that the system is concerned about the prospect of the box being shaken as either part of an attack or an earthquake (i.e, accelerometer $\in M$, shaking $\in I$, and attacker and earthquake $\in E$). However, because shaking is not enough to differentiate between the two events, the attack and earthquake event nodes are combined into a single node in the TIP's limited factor graph.

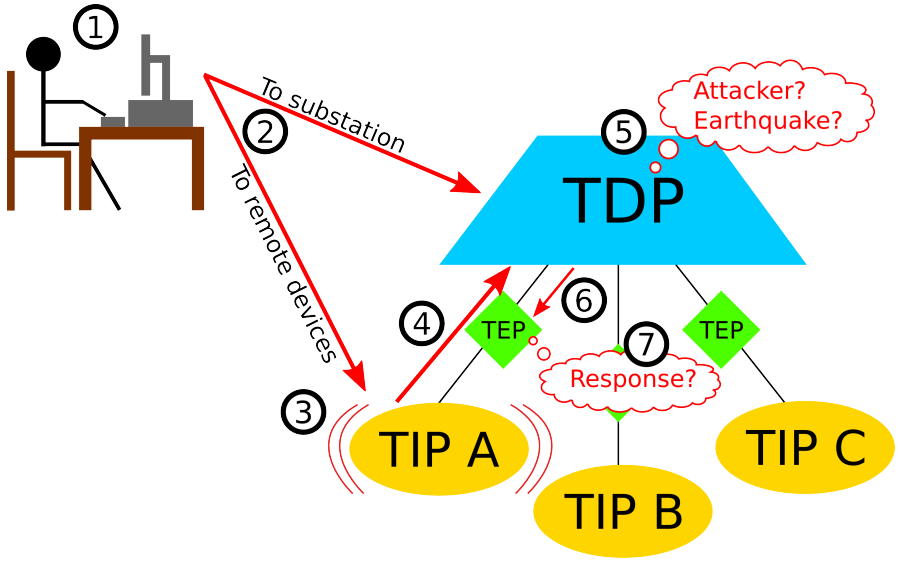

Figure 3: A diagram of the example given in Section 4.4. First, a utility operator builds the tamper system using the generation tool (Step 1), and then deploys the various components to their proper locations in the network (Step 2). When a TIP senses shaking (Step 3), it sends an alert to the TDP (Step 4), which then uses it full information base to decide exactly what is happening (Step 5). This decision is then sent to the appropriate TEPs (Step 6), who then decide the proper response to the event (Step 7).

2. The TDP and TIPs programs are generated and deployed, and the TIPs begin monitoring their environment for event indicators.

3. TIP A experiences intense shaking, causing its accelerometer to exceed its threshold. A's limited factor graph determines that its attack/earthquake event is occurring, but it cannot differentiate between the two possibilities without knowing the state of other boxes in the system.

4. A sends an alert to the TDP about its situation, and request assistance on making a decision.

5. The TDP receives A's alert and attempts to determine the overall state of the system. In doing so, it finds that B and C have also experienced intense shaking (and in turn, have also sent the TDP alerts about potential tampering).

6. The widespread shaking causes the TDP's factor graph to decide that the shaking is due to an earthquake. This decision is passed along to the TEP associated with A. (Since B and C also sent alerts, their TEPs would also be alerted.)

7. A's TEP weighs its options, and decides not to take any action for the time being, since the earthquake does not put the SCADA network at risk and the utility wants to keep the system running as long as possible.

5 Action Plan

Currently, we have built a simple prototype of our tamper protection system in which a TDP takes information from its TIPs and makes event decisions when asked. While we have tested our prototype on standard Linux desktop systems, we have not yet tested T.E.D.D.I. on power devices within a representative SCADA network.

Our future plan for this project is as follows:

1. Add tamper enforcement points to our current prototype, and finalize our algorithm for calculating the optimal response to an event.
2. Begin building the T.E.D.D.I. generation tool, and include all of the features mentioned in Section 4.3. We will use our current prototype as a template for exactly how the programs should be coded.
3. Evaluate both the speed and accuracy of our system on several non-trivial SCADA network implementations, which will ideally be drawn from actual distribution network topologies provided by partner utilities.
4. Finally, we will install our programs on actual power hardware on a simulated network (such as the TCIPG Testbed [36]) to see how our system behaves in a more realistic scenario, and whether our correctness and performance claims hold up in this environment. We would also like to get our tool into the hands of power industry personnel to evaluate the usability of the tool.

6 Conclusions

In this paper, we identified the security threat introduced by remotely-deployed embedded devices within the power grid, and proposed T.E.D.D.I. as a way to address these threats and protect SCADA networks within the grid. We showed how the use of factor graphs overcomes the shortcomings of existing work in the tamper detection and SCADA protection spheres, and demonstrated how our tamper information, decision, and enforcement points work together to detect important events and mitigate their effects. While T.E.D.D.I. is only a prototype at this point, we outlined a future plan to build a tool that utilities can use to generate a tamper protection system for an arbitrary network, and to evaluate the performance of T.E.D.D.I. on representative SCADA networks.

The transition to a smarter electric grid is a difficult one, as it increases the attack surface that utilities must protect. With T.E.D.D.I., we hope to help utilities better manage their security risks, and thus reap the benefits of a modernized grid infrastructure.

Acknowledgements

The authors would like to thank Ryan Bradetich, Jason Dearien, Dennis Gammel, and Rhett Smith of Schweitzer Engineering Laboratories, Elaine Palmer of IBM Watson, Steve Weingart of Aruba Networks, and Bill Nisen of the Institute for Security, Technology, and Society (ISTS) for their support and suggestions on this project.

This material is based upon work supported by the Department of Energy under Award Number DE-OE0000097. Portions of this work are based on the thesis proposal of the first author.

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process,

or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

References

[1] IEEE standard communication delivery time performance requirements for electric power substation automation. IEEE Standard 1646-2004. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1405811.

[2] ACE 3600 specifications sheet. http://www.motorolasolutions.com/web/Business/Products/SCADA%20Products/ACE3600/_Documents/Static%20Files/ACE3600%20Specifications%20Sheet.pdf.

[3] Atmel trusted platform module. http://www.atmel.com/products/security-ics/embedded/default.aspx.

[4] Robin Berthier and William Sanders. Monitoring advanced metering infrastructures with Amilyzer. In *Cybersecurity of SCADA and Industrial Control Systems (C&ESAR)*, 2013.

[5] Avinash Desai. Anti-counterfeit and anti-tamper implementation using hardware obfuscation. Master's thesis, Virginia Polytechnic Institute and State University, August 2013.

[6] Silvio Dragone. Physical security protection based on non-deterministic configuration of integrated microelectronic security features. In *The First International Cryptographic Module Conference*, September 2013.

[7] Organization for the Advancement of Structured Information Standards. *eXtensible Access Control Markup Language (XACML) Version 3.0*, January 2013. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf.

[8] Brendan Frey. Extending factor graphs so as to unify directed and undirected graphical models. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, 2003.

[9] HP trusted platform module. http://h18004.www1.hp.com/products/servers/proliantstorage/module.html.

[10] IBM 4765 PCIe data sheet. http://www-03.ibm.com/security/cryptocards/pciecc/pdf/PCIe_Spec_Sheet.pdf.

[11] Vadim Indelman, Stephen Williams, Michael Kaess, and Frank Dellaert. Information fusion in navigation systems via factor graph based incremental smoothing. *Robotics and Autonomous Systems*, 61(8):721–738, 2013.

[12] Intel trusted platform module hardware user's guide. http://download.intel.com/support/motherboards/server/sb/g21682004_tpm_hwug1.pdf.

[13] Stephen Kent. *Protecting Externally Supplied Software in Small Computers*. PhD thesis, Massachusetts Institute of Technology, September 1980.

[14] Hans-Andrea Loeliger, Justin Dauwels, Junli Hu, Sascha Korl, Li Ping, and Frank R. Kschischang. The factor graph approach to model-based signal processing. *Proceedings of the IEEE*, 95(6):1295–1322, 2007.

[15] Rajesh Kannan Megalingam, Ashok Krishnan, Bharath Kalathiparambil Ranjan, and Amar Kelu Nair. Advanced digital smart meter for dynamic billing, tamper detection, and consumer awareness. In *Proceedings of the 3rd International Conference on Electronics Computer Technology*, 2011.

[16] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kauffman, 1988.

[17] Jason Reeves. Autoscopy Jr.: Intrusion detection for embedded control systems. Master's thesis, Dartmouth College, September 2011. Revised version of August 2011 thesis submission.

[18] Christopher Roblee, Vincent Berk, and George Cybenko. Large-scale autonomic server monitoring using process query systems. In *IEEE International Conference on Autonomic Computing*, 2005.

[19] SEL-651R advanced recloser control. `https://www.selinc.com/SEL-651R/`.

[20] SEL-651R-2 recloser control data sheet. `https://www.selinc.com/WorkArea/DownloadAsset.aspx?id=100135`.

[21] Rebecca Smith. U.S. risks national blackout from small-scale attack, March 2014. `http://online.wsj.com/news/articles/SB10001424052702304020104579433670284061220`.

[22] Sean W. Smith, Elaine Palmer, and Steve Weingart. Using a high-performance, programmable secure coprocessor. In *Second International Conference on Financial Cryptography*, 1998.

[23] Sean W. Smith and Steve Weingart. Building a high-performance, programmable secure coprocessor. *Computer Networks*, 31(1999):831–860, 1999.

[24] Rouslan V. Solomakhin. Predictive YASIR: High security with lower latency in legacy SCADA. Master's thesis, Dartmouth College, June 2010.

[25] William L. Sousan, Quiming Zhu, Robin Gandhi, and William Mahoney. Smart grid tamper detection using learned event patterns. In Vijay Pappu, Marco Carvalho, and Panos Pardalos, editors, *Optimization and Security Challenges in Smart Power Grids*, Energy Systems, pages 99–115. Springer Berlin Heidelberg, 2013.

[26] George Theodorakopoulos and John S. Baras. On trust models and trust evaluation metrics for ad hoc networks. *IEEE Journal on Selected Areas In Communications*, 24(2):318–328, 2006.

[27] J. Doug Tygar and Bennet Yee. Dyad: A system for using physically secure coprocessors. In *Technological Strategies for the Protection of Intellectual Property in the Networked Multimedia Environment*, 1994.

[28] Alfonso Valdes and Keith Skinner. Probabilistic alert correlation. In *Recent Advances in Intrusion Detection*, 2001.

[29] Elizabeth Von Meier. *Electric Power Systems: A Conceptual Introduction*. John Wiley and Sons, Inc., 2006.

[30] Yujue Wang and Carl Hauser. An evidence-based Bayesian trust assessment framework for critical-infrastructure decision processing. In *Fifth Annual IFIP Working Group 11.10 International Conference on Critical Infrastructure Protection*, 2011.

[31] Gabriel Weaver, Carmen Cheh, Edmund Rogers, William Sanders, and Dennis Gammel. Toward a cyber-physical topology language: Applications to NERC CIP audit. In *ACM Workshop on Smart Energy Grid Security (SEGS '13)*, 2013.

[32] Steve Weingart. Physical security devices for computer subsystems: A survey of attacks and defenses 2008 (updated from the CHES 2000 version), 2008. Originally from *Second International Workshop on Cryptographic Hardware and Embedded Systems*, August 2000.

[33] A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, and S. Waldbusser. *Terminology for Policy-Based Management (RFC 3198)*. Internet Engineering Task Force, November 2001. `http://www.ietf.org/rfc/rfc3198`.

[34] Steve White and Liam Comerford. ABYSS: A trusted architecture for software protection. In *IEEE Symposium on Security and Privacy*, 1987.

[35] Steve White, Steve H. Weingart, William Arnold, and Elaine Palmer. Introduction to the Citadel architecture: Security in physically exposed environments. Technical Report RC16672, IBM T. J. Watson Research Center, 1991.

[36] Tim Yardley, Jeremy Jones, David Nicol, and William Sanders. Testbed overview. Fact Sheet from the 2013 TCIPG Industry Workshop, 2013.

[37] Saman Zonouz, Himanshu Khurana, William Sanders, and Timothy Yardley. RRE: A game-theoretic intrusion response and recovery engine. *IEEE Transactions on Parallel and Distributed Systems*, 25(2):395–406, 2014.

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