Resiliency-Aware Deployment of SDN in Smart Grid SCADA: A Formal Synthesis Model

A H M Jakaria*, Mohammad Ashiqur Rahman†, and Aniruddha Gokhale‡
*Department of Computer Science, Tennessee Tech University, Cookeville, USA
†Department of Electrical and Computer Engineering, Florida International University, Miami, USA
‡Department of Computer Science, Vanderbilt University, Nashville, USA
Email: *ajakaria42@students.tntech.edu, †marahman@fiu.edu, ‡a.gokhale@vanderbilt.edu

***Abstract*—The supervisory control and data acquisition (SCADA) network in a smart grid requires to be reliable and efficient to transmit real-time data to the controller, especially when the system is under contingencies or cyberattacks. Introducing the features of software-defined networks (SDN) into a SCADA network helps in better management of communication and deployment of novel grid control operations. Unfortunately, it is impossible to transform the overall smart grid network to have only SDN-enabled devices overnight because of budget and logistics constraints, which raises the requirement of a systematic deployment methodology. In this paper, we present a framework, named SDNSynth, that can design a hybrid network consisting of both legacy forwarding devices and programmable SDN-enabled switches. The design satisfies the resiliency requirements of the SCADA network, which are determined with respect to a set of pre-identified threat vectors. The resiliency-aware SDN deployment plan primarily includes the best placements of the SDN-enabled switches (replacing the legacy switches). The plan may include one or more links to be installed newly to provide flexible or alternate routing paths. We design and implement the SDNSynth framework that includes the modeling of the SCADA topology, SDN-based resiliency measures, resiliency threats, mitigation requirements, the deployment budget, and other constraints. It uses satisfiability modulo theories (SMT) for encoding the synthesis model and solving it. We demonstrate SDNSynth on a case study of an example small-scale network. We also evaluate SDNSynth on different synthetic SCADA systems and analyze how different parameters impact each other. We simulate the SDNSynth suggested networks in a Mininet environment, which demonstrate the effectiveness of the deployment strategy over traditional networks and randomly deployed SDN switches in terms of packet loss and recovery time during network congestions.**

***Index Terms*—SDN architecture; incremental deployment; smart grid; SCADA; formal modeling; network synthesis**

I. Introduction

SMART grids are large, heterogeneous, and distributed in nature, where the maintenance of a large number of intelligent end devices is required. This task is complex and requires careful management. The SCADA network infrastructure of a smart grid needs to be reliable and efficient to transmit a large amount of real-time data [1]. The observability of a grid bus system is determined by the successful delivery of critical measurements collected by the end devices to control centers. The overall network should be resilient to cyberattacks to ensure seamless transmission of control and measurement data from all the devices that provide sensitive data to retain system observability. However, the end devices and the network forwarding devices can get compromised, which leads to severe quality issues in the smart grids.

Current network infrastructures in SCADA systems use diverse protocols and heterogeneous forwarding devices [2]. These protocols and devices make the management, maintenance, and integration of new devices difficult. Software-defined networking (SDN) has great potential to be used in SCADA systems [3]. It not only provides flexibility to implement novel networking solutions and quality of service (QoS) optimization but also provides greater resiliency to cyberthreats [4], [5].

However, the primary challenge for solutions built on SDN in an enterprise network is the deployment problem. Network upgrades in any enterprise are budget and resource-constrained. It is impractical to substitute all the legacy switches with SDN switches overnight. The process of simultaneous deployment and operation of legacy and SDN-enabled switches remains one of the greatest challenges in incorporating SDN to smart grids. The already deployed traditional switches and routers need to be replaced with SDN switches systematically, so that the hybrid network can still be benefited by the features of SDN.

The problem of deployment of SDN satisfying grid observability constraints within a limited budget is a recent topic and is generally an NP-hard one [6], [7]. Utilizing the available limited budget (*e.g.*, a limited number of SDN-enabled switches), while perceiving the benefits of SDN, is challenging. We propose to formally model the constraints and requirements into a constraint satisfaction problem (CSP) and solve it using a CSP solver to generate the SDN-enabled network architecture for the SCADA. We present an automated framework, SDNSynth, which solves this problem using formal verification.

We aim to resolve security challenges by providing resiliency to the SCADA network. We use vulnerable sets of electronic devices that form various threat vectors (as identified in [8]). The network can be resilient to attacks if proper communication between the devices and the control center can be ensured, despite attacks, such that the center receives necessary measurements to observe the complete system . The observability can be ensured through secure and alternative communication of the remote devices utilizing the benefits of SDN. However, the available SDN switches are required to

be properly placed so that it is possible to reroute control and data traffic and set up virtual networks whenever needed.

SDNSynth provides an automated tool for synthesizing SDN switch placements, as well as topology extension through new link deployment, using constraint satisfaction checking. It takes the existing network topology, security requirements, and physical resources as inputs and formulates the deployment problem. We solve the problem by encoding the model into first-order logic. We use SMT in the encoding purpose, which also provides us a solution if there is any, in the form of the deployment plan for SDN switches. The major contributions of this paper include:

1) Formal modeling of the resiliency requirements, resource constraints, and network topology that implements SDN.
2) Designing a framework that can efficiently generate the candidates for SDN switch placements (replacing the legacy switches) and new link deployments (topology extension) that improve the resiliency of the SCADA network by satisfying all the requirements and constraints. To the best of our knowledge, this work is the first of its kind that designs the SDN topology with respect to the observability analysis of smart grid SCADA systems.
3) Implementing a tool based on the framework and a thorough evaluation of its performance.
4) Simulating the network suggested by the tool in a Mininet-based virtual environment.

We briefly introduced SDNSynth in [9]. Although the research considers SCADA as the problem domain, the process of simultaneous deployment and operation of legacy and SDN-enabled switches remains as one of the greatest challenges in general. The proposed framework is generic enough to be used for SDN deployment problems in cyber and other cyber-physical systems (CPSs).

The rest of this paper is organized as follows: Section II presents an overview of SDN, its impact on smart grids, and the related works. We discuss the framework of the proposed solution in Section III. Section III also describes the formal model, while the implementation and a case study are discussed in Section IV. The evaluation results of our model are presented in Section V. Section VI concludes the paper.

II. Background and Research Objective

A smart grid is a combination of power grids, communication networks, and information management systems. The SCADA system is a core component of a smart grid. It consists of heterogeneous smart devices, such as intelligent electronic devices (IED), programmable logic controllers (PLC), remote terminal units (RTU), master terminal units (MTU), control servers, routing and security devices, etc [10]. These devices communicate with each other under various communication protocols, physical media, and security properties. A SCADA network analyzes real time data about power generation, flow, distribution, and consumption in power grid systems. The controllers use this analysis to maintain critical energy management operations, such as state estimation, automatic generation control, etc.

A. SCADA Security Issues

In a SCADA network, security can become an issue in many different ways. The following includes some examples:

- In case several critical smart devices such as IEDs, RTUs or PLCs get compromised, the overall network becomes vulnerable to cascading attacks [11]. The system may not be observable if a critical set of measurements cannot reach the control center.
- Forwarding devices (routers/switches) can be compromised in a stealthy way. In these cases, the attack might not be detected and cannot be mitigated easily. This yields to unwanted packet delay, which can lead to synchronization issues and performance degradation of overall controls [12]. This can also bring down routes in dedicated networks and create congestion, which ultimately results in valid packet drops.
- False command and other data injection is another security issue in SCADA [13]. This can cause incorrect system estimations and control decisions, which leads to infrastructure damages, as well as power outages.

B. Software-Defined Networking

In a data communication network, many hosts share a network infrastructure, which consists of many routing or switching devices. Most of the time, these routing devices are closed systems and have very limited or vendor-specific configuration interfaces. Once deployed and in production, it is very hard to reconfigure the network and introduce novel applications or new versions of existing protocols. SDN has been developed to decouple the data plane in a network infrastructure from the control plane [14]. A programmatic controller performs the logic to reroute and manage the traffic by communicating with simple forwarding devices, known as SDN-enabled switches. When a packet arrives at such a switch, it extracts some packet header fields and matches against some flow-table entries. If a match is found, the packet is forwarded or dropped according to the action specified in the entry. Otherwise the packet or some features of the packet are sent to the controller. The controller instructs the switch to add a new flow-table entry and perform forwarding accordingly. This is made possible by a protocol named 'OpenFlow' [15]. This mechanism reduces the network effectively to consisting of simple forwarding hardware devices and one or a few decision-making controllers.

C. SDN and SCADA Resiliency

SDN can impact the SCADA resiliency mainly by its ability to perform traffic engineering, which cannot be done with legacy switches or routers flexibly. For example:

- Rerouting data, whenever the system is required to bypass a router/device because it is compromised, is made easier by SDN. During the times of unavailability of normal routes due to failure or attack, fast rerouting is possible utilizing SDN, which allows controllers to take routing decisions more quickly than traditional routing algorithms. Alternate paths for data ensure greater resiliency to the grid system [16].

- SDN can be used to establish alternate or dynamic paths for grid control commands and other data from the control center. The commands can be transmitted from a control center to grid devices exactly at the required moments. This approach significantly reduces the time-window in which the attacker can inject malicious commands [17].
- If there are any compromised switches or control devices, it is easier for SDN to isolate the nodes from the routing topology through the usage of VLANs. In this way, the network of intelligent devices in the smart grid remains resilient to cyberthreats.
- In a smart grid, there are different types of data and control flows that have different levels of criticality [18]. SDN can also dynamically prioritize the packet flows in the case of congestion/denial of service attacks in the network. SDN can impose rules on the switches to set up a priority for relaying these packets efficiently [19].
- SDN allows switching to public network with proper encryption, when the dedicated network for SCADA is unavailable [17].

D. Research Objectives

In this paper, we aim to answer the following questions: *1) Where and how many SDN switches and links should be deployed to achieve specific traffic engineering goals given budget and resource constraints? 2) Is there a trade-off between the cost of SDN deployment and system resiliency? 3) Is the bus system observable after deploying the SDN components, even in the cyberattack scenarios to some extent?*

To answer these, we devise a tool to strategically select a subset of legacy devices in an existing SCADA network and replace them with SDN. The research addresses the following many-fold objectives: 1) upgrade the existing network with SDN-enabled switches within the available budget; 2) synthesize the SDN topology with newly available links within the budget, which enhances network resiliency; 3) achieve successful and resilient observability, given sufficient measurement delivery from IED to MTU, which leads to successful state estimation.

E. Related Works

There are several works available that introduce the usage of SDN techniques in smart grids. Chekired *et al.* presented a model for energy management in smart grids based on cloud-SDN architecture [20]. Jin *et al.* and Ren *et al.* presented the impact of SDN on microgrid resiliency [21], [22]. Cahn *et al.* proposed that software-defined networking can alleviate many of today's problems regarding setup complexity to security policies in energy communication networks (ENC) and create a network, which can evolve with changing technologies and needs [3]. Zhang *et al.* presented three use cases to examine the opportunities for SDN technology in smart grid [4]. These use cases include enhancing data exchange, virtual network for distributed energy resources aggregation, and smart building energy management. Feamster *et al.* clarified the relationship of SDN with several common network technologies such as network virtualization [23].

An SDN-based approach to modernize the SCADA systems was investigated by Da Silva *et al.* [24]. The authors simplified the management of power system resources, as well as designed a mechanism to prevent eavesdroppers from capturing communication flows between SCADA components. In [25], the authors proposed an intrusion detection system that utilizes SDN and characteristics of SCADA for traffic classification. Dong *et al.* investigated how SDN can enhance the resilience of typical smart grids to malicious attacks. They also identified additional risks introduced by SDN and how to manage them [17]. However, none of the above works discuss the gradual deployment plan of SDN-enabled switches in a hybrid network scenario.

A few works are available for the incremental deployment of SDN in enterprise networks. One of the most typical solutions is to divide the data flows through two different domains: one is SDN switches and another is legacy switches. The main problem with this approach is that some packets get treated by the SDN controller via the SDN switches, while some others do not. This approach prevents an enterprise to realize the ultimate benefits of SDN. Several manual heuristic based algorithms have been devised to determine the locations of the limited number of SDN switches [6]. Hong *et al.* systematically studied the incremental SDN deployment problem by formulating it as an optimization problem and proposed effective heuristics for selecting a small set of existing devices for upgrading [6]. They formulated the SDN deployment problem as an optimization problem and showed that it is still NP-complete. Hence, they developed several simple but effective heuristics to tackle it.

An incremental deployment strategy using a heuristic algorithm was proposed by Xu *et al.* [7]. The authors aimed for throughput maximization in routing. They applied a depth-first-search method and a randomized routing mechanism to solve the routing problem in a hybrid SDN environment.

A genetic algorithm was developed by Guo *et al.* to determine the sequence of migration of legacy routers to SDN, which provides the most benefits from the perspective of traffic engineering [26]. Their approach minimized the total maximum link utilization, which is subject to constraints such as link capacity, total amount of incoming and outgoing flows on a node, etc. They demonstrated that this algorithm performed better than a greedy and static approach that was also proposed by them.

Levin *et al.* presented the design and implementation of an architecture called Panopticon for operating networks that combine legacy and SDN switches [27]. Panopticon exposes an abstraction of a logical SDN in a partially upgraded legacy network, where SDN benefits can extend over the entire network. They also formalized an optimal cost-aware upgrade algorithm based on mathematical programming [28]. A solution for seamless peering between SDN and existing IP networks with the deployment of new SDN features was studied by Jonathan *et al.* [29]. Effective use of SDN for traffic engineering, while SDN is incrementally being deployed, was discussed by Agarwal *et al.* [30]. Das *et al.* used a slicing pane between the SDN switches and the controller to partition the data plane into multiple slices which are controlled by different

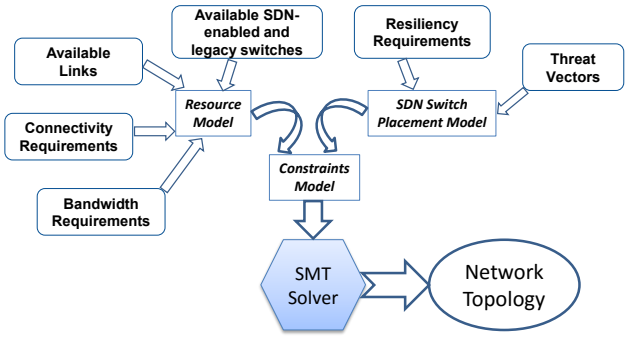


Fig. 1. The framework architecture of SDNSynth.

controllers [31]. This also helped in the gradual deployment of SDN.

Poularakis *et al.* study the gradual upgrade of an ISP network to include SDN-enabled switches [32]. They focused on maximizing the traffic flowing through the SDN-switches and the number of dynamically selectable alternate paths while considering the cost. They do not consider the criticality of the sources of traffic, which is a key factor in our study.

None of these existing works solve the problem from the smart grid observability point of view. They do not solve the multi-objective NP-hard problems of generating an SDN topology with grid resiliency in mind. We aim to solve the deployment problem while maintaining the resiliency of the SCADA network.

III. Synthesis of SDN Deployment Network

This section first discusses the framework of our proposed resiliency-aware deployment model. It also provides the formal modeling of the requirements and the constraints of the SDN-enabled network architecture. Table I lists several variables used in the model. It is notable that no multiplication of two parameters is performed in the paper without using the multiplication sign.

A. SDNSynth Framework

In energy management system (EMS), optimal power flow (OPF) is dependent on a core routine called 'state estimation' [33], [34]. A smart grid computes currents, voltages, phase angles, transmission line power flow, loads at the buses, etc. [35]. These measurements are used to estimate a number of power system state variables. A Jacobian Matrix represents the relationship between the measurements and the state variables [36]. The state estimation can be vulnerable to false data injection, or the unavailability of required data, in the case of an adversary being able to alter or eliminate some measurements without being detected [37].

Rahman *et al.* proposed a mechanism for automatic synthesis of a secure set of measurements and intelligent devices, with respect to a list of security requirements (*i.e.*, expected attack model [8]). Their framework can identify the most critical measurements, based on a given set of attacker capability. From this work, we will use threat vectors, which are basically sets of IEDs, as our input to SDNSynth. The devices in the threat vectors are related to vulnerable measurements and are considered to be critical. SDNSynth ensures the delivery of measurements from these critical end devices by placing SDN-enabled switches on their paths to the control center. The security and resiliency requirements, considered in this work, will ensure that a SCADA control process receives sufficient data (*i.e.*, measurements from field devices) to perform its operation even in limited contingencies. We ensure the observability analysis, which is a prior and crucial requirement for state estimation of the power system control routine [38].

SDNSynth follows a top-down architecture design automation approach instead of the traditional bottom-up approach. The major features of SDNSynth are as follows:

- Formally models the network topology, required configurability (*i.e.*, SDN features) of switches by the controller, and resource constraints.
- Formalizes the incremental SDN design synthesis problem as the determination of deployment decision of SDN switches, new links, and their placements that satisfy the given requirements and constraints.
- Encodes the synthesis problem into SMT logics and provides a feasible solution using an SMT solver and following a hill-climbing approach.

The SDNSynth architecture is shown in Fig. 1. SDNSynth takes the following as inputs: (i) security requirements and threat vectors, (ii) the network topology including bandwidths of the links, which constitute the observability model of the network, and (iii) available SDN switches and links that can replace the traditional switches, which make up the resource model. The tool takes input from a user using an input file. The output of the tool indicates the best possible candidates for switch replacements, as well as the new links that should be deployed in the network according to availability and budget.

SDNSynth will be run by network operators whenever they have adequate budget for the deployment of new SDN switches. The tools takes all the existing switches/routers (both SDN-enabled and traditional) into account, and produces a satisfiable solution, if any, satisfying the resiliency requirements.

B. Preliminary of Formal Model

While designing a hybrid topology consisting of SDN-enabled switches, we need to ensure the communication of critical devices (*i.e.*, IEDs) with the MTU by ensuring alternate paths from them to the MTU. SDN switches can be instructed by the controller to change the behavior of routing based on certain scenarios. For example, if a route fails because of some router/device/link failure, there can be congestion in the available routes. In these cases, we need to ensure the traffic from the most critical devices, instead of all the devices. Distinguishing between traffic from different sources is much easier with programmable switches than traditional routers [39]. If the IEDs themselves are down or compromised and an attacker can find sensitive data through it, we can stop the flow of any data to and from it by instructing the SDN

TABLE I
NOTATION TABLE

Notation	Definition
$Node_i$	Whether node (IED) i is available.
$\mathbb{P}_i$	The set of all possible paths from IED i to MTU.
$p_{i,y}$	The y^{th} path from IED i to the MTU.
$d_{i,y}$	Set of all forwarding devices on y^{th} path from IED i.
$Link_l$	If link l is up.
$SwitchIsSDN_s$	If switch s is SDN-enabled.
$AssuredAltPath_i$	Whether there are alternate paths from IED i to MTU.
$AssuredBandwidth_i$	Whether adequate bandwidth is assured for IED i.
$AssuredDelivery_i$	Whether data delivery is assured from IED i to the MTU.
$VlanEnablingSwitch_s$	If switch s can create a VLAN.

switches. We also need to ensure the adequate bandwidth for each link in the communications paths, so that data from all end devices sharing the path are not clogged up. We aim to also retrieve a deployment plan for the newly available links.

Throughout the model of the network synthesis, for simplicity, we have used the sum of boolean variables to formally model several constraints. While encoding the summation of boolean variables, we have created corresponding integer variables (0 or 1) and used their sum for calculation.

C. Observability

The power system is observable when the measurements collected by IEDs can solve a list of unknown state variables. The observability of a smart grid ensures three conditions: (i) the unique measurements collected by the IEDs are successfully delivered to the MTU, (ii) the measurements can cover all the unknown state variables, and (iii) the number of these measurements is greater than or equal to the number of variables [8].

$k-$Resilient Observability: This constraint verifies that if k field devices (*e.g.*, IEDs) are unavailable, whether observability is still ensured. A device is unavailable when it fails to communicate with the MTU due to its technical failure or remote attacks on it or on the communication path toward the destination. We can find out if there is a set of devices, no more than k in number, which can make the system unobservable when they are unavailable. This set is a threat vector that entails that the system is not $k-$resilient. If there is no such threat vector, then it is $k-$resilient observable [8]. If $Node_i$ denotes whether node i (an IED) is available, a threat vector ($\mathbb{V}$) represents the set of devices for which the following statement is true in the case of unobservability: $\forall_{i\in\mathbb{V}}\neg Node_i$. In other words, if all the devices (IEDs) in a threat vector are unavailable simultaneously, the system will be unobservable.

Let N be the number of end devices. The $k-$resilient observability constraint ($\neg ResilientObservability$) can be formalized as follows:

$$((N - \sum_{1\leq i\leq N} Node_i) \leq k) \wedge \neg Observability \rightarrow \neg ResilientObservability$$

After finding out the threat vectors, we need to make sure that the communication path from at least one device in each threat vector is ensured in the case of an attack. The user can specify the required percentage of threatened devices that need to be protected. The communication between a field device (*i.e.*, IED or RTU) and the MTU is dependent on the communication routes (links) and the intermediate devices (RTUs or routers). We need to ensure that there are adequate number of alternate paths from these IEDs to the MTU. Our goal is to secure the paths taken by these critical measurements by suggesting alternate paths. SDN switches ensure a better way to reconfigure the network in the case of a switch/router failure or network congestion. If SDN switches can be deployed on the paths to ensure alternate paths for critical measurements to be delivered from IEDs to MTU, the network would be much more resilient to threats.

D. Priority Management

Algorithm 1 Priority Assignment Algorithm

```
1: for each IED i do
2:     rank[i] := ω * NumSensorActuator(i) + ω̄ * Σ_v IsInVector(i, v)
3:     priority[i] := 0      ▷ initialize priorities of all IEDs with 0.
4: end for
5: maxRank := Max(rank)
6: chunkSize := maxRank/numPriorityType
7: for each priority type p do
8:     start := p × chunkSize.
9:     end := (p = numPriorityType) ? maxRank : (start + chunkSize)
10:    for each IED i do
11:        if InRange(rank[i], start, end) then
12:            AssignPriority(priority[i], p)
13:        end if
14:    end for
15: end for
16: return priority
```

In a SCADA environment, there are some smart devices such as IEDs that collect data from the grid. These may perform the collection with the help of sensors such as ammeter, flowmeter, etc. The data is forwarded to RTUs or PLCs depending on the type required. The RTUs send the data to the central control center or the MTU. In the other direction, the control center sends necessary commands to the IEDs with the help of RTUs. Some IEDs are connected to actuators such as a circuit breaker. The actuators perform the changes that are required in the grid. The IEDs that have sensors or actuators with them are of high priority. Any communication involving them must be ensured in order to maintain the grid operations without any error.

Another criterion for measuring the criticality of devices is how vital their collected data is. There are hundreds of thousands of measurements collected by smart devices. Some IEDs are less critical as the measurements they collect are

not unique, *i.e.*, some other IEDs report the same or related measures. Hence, they are not individually crucial to estimate the corresponding state variable. The IEDs collecting unique measurements are the most critical ones as, without these measurements, state estimation cannot solve for some state variables. Hence, the criticality of an IED inversely depends on the extent of redundancy (*i.e.*, alternatives). These critical IEDs are more prominent in the successful attack vectors [8]. It is essential to ensure that any critical device must be able to communicate with others, as long as they are not compromised.

In Algorithm 1, we devise a method to determine the criticality/priority of the field devices. The rank of an IED increases if it has some attached sensors or actuators. The rank also increases if it is an element of one or more sets of threat vectors. ω is the impact of the number of sensors/actuators in calculating the rank of the IEDs, while $\overline{\omega}$ is the impact of the existence of an IED in the threat vectors. These have values between 0 and 1, which impact the calculation of the ranks of the IEDs. The rank is used in calculating the priorities of the IEDs. The algorithm returns the priorities of the IEDs according to the number of priority types specified by the user. For example, we can define three levels of priorities for the IEDs: high, medium, and low. When modeling the resiliency requirements of the devices, the order of the devices follow the priority of them. That means, the devices in the threat vectors having higher priority will be ensured more alternate paths and other resiliency features before others.

E. Resiliency Management

Here we present our model according to several SDN benefits in the management of security and resiliency of SCADA networks.

Alternate Paths: First, we define alternative paths for the IEDs. In calculating alternate paths from an IED to an MTU, we consider α-Alternative paths, where α means the percentage of overlapped/shared links on the paths. If a path contains less than $\alpha\%$ of common links with another path, it can be considered as an alternate to the other one.

We consider all possible forwarding paths from an IED i to the MTU, through one or more RTUs, as $\mathbb{P}_i$. A path $p_{i,y}$ is defined as the y^{th} path among all possible paths. $p_{i,y}$ is a set of links, while each link l represents a pair of nodes that belong to the set $\mathbb{L} \subseteq \mathbb{N} \times \mathbb{N}$, assuming that $\mathbb{L}$ is the set of links and $\mathbb{N}$ is the set of all nodes.

Let $\mathbb{I}$ be the set of all IEDs in the threat vectors. If $AltPath_{p_{i,y},p_{i,y'}}$ denotes that path $p_{i,y'}$ is an alternate path for $p_{i,y}$, then:

$$\forall_{i\in\mathbb{I}}\forall_{p_{i,y},p_{i,y'}\in\mathbb{P}_i} AltPath_{p_{i,y},p_{i,y'}} \rightarrow (\sum_{l\in p_{i,y}} l \in p_{i,y'}) \leq (|p_{i,y'}|\times\alpha) \quad (1)$$

Next, we find the switches that split a path of an IED to create multiple alternate paths. If $\mathbb{S}$ is the set of all candidate switches to be replaced by SDN-enabled ones and $\mathbb{L}_s$ is the set of all links connected to switch s, then $SwitchOnAltPathBranch_s$ ensures that switch s is positioned where two or more alternate paths for IED i branches.

$$SwitchOnAltPathBranch_s \rightarrow AltPath_{p_{i,y},p_{i,y'}} \wedge \sum_{l,l'\in\mathbb{L}_s} \big((l \in p_{i,y}) \wedge (l' \in p_{i,y'})\big) \geq 1 \quad (2)$$

SDN-Enabled Switches: SDN-enabled switches should be deployed intelligently on the network branches. We want the SDN-enabled switches to be deployed on the alternate paths for the IEDs. Also, they should be deployed at the forks of the paths, so that SDN controller is able to route the data and command packets to and from IEDs efficiently, and according to priority. We define such alternate paths as software-defined alternate paths, $SDAltPath_{p_{i,y},p_{i,y'}}$. If $SwitchIsSDN_s$ denotes whether switch s is SDN-enabled or not, $d_{i,y}$ is the set of all switches on the y^{th} path from IED i to the MTU, and $d_{i,y'}$ is the set of all switches on path y', the following should hold:

$$\forall_{i\in\mathbb{I}}\forall_{p_{i,y},p_{i,y'}\in\mathbb{P}_i} SDAltPath_{p_{i,y},p_{i,y'}} \rightarrow AltPath_{p_{i,y},p_{i,y'}} \wedge \exists_s (s \in d_{i,y}) \wedge (s \in d_{i,y'}) \wedge SwitchIsSDN_s \wedge SwitchOnAltPathBranch_s \quad (3)$$

$$\forall_{i\in\mathbb{I}}\ AssuredMinAltPath_i \rightarrow (Priority_i = m) \wedge \forall_{p_{i,y}} (1 + \sum_{y'} SDAltPath_{p_{i,y},p_{i,y'}} \geq minAltPath_m) \quad (4)$$

If m is the priority of IED i obtained from Algorithm 1, in the Equation 4, $minAltPath_m$ is a constant specifying the minimum number of alternate paths that need to exist between IED i and the MTU. Each path $p_{i,y}$, for an IED i with priority m, should have at least $minAltPath_m$ alternate paths $p_{i,y'}$.

Different number of alternate paths may be specified for different levels of priorities. For example, at least five alternative paths should be deployed for high priority communications, while low priority devices may require at least two alternate paths, for each of its paths to the MTU.

All the links in the communication paths and their alternate paths need to be deployed and up. If $Link_l$ is a boolean variable denoting whether a link is up or not,

$$\forall_{i\in\mathbb{I}} AssuredLinksOnAltPath_i \rightarrow SDAltPath_{p_{i,y},p_{i,y'}} \wedge \forall_{l\in p_{i,y}} Link_l \wedge \forall_{l'\in p_{i,y'}} Link_{l'} \quad (5)$$

$AssuredMinAltPath_i$ and $AssuredLinksOnAltPath_i$ ensure the existence of alternate paths from IED i to the MTU.

$$\forall_{i\in\mathbb{I}} AssuredAltPath_i \rightarrow AssuredMinAltPath_i \wedge AssuredLinksOnAltPath_i \quad (6)$$

If $altPathExp$ denotes the expected percentage of IEDs to have assured alternate paths, then the following should hold:

$$\frac{\sum_i AssuredAltPath_i}{|\mathbb{I}|} \geq altPathExp \quad (7)$$

Bandwidth Requirements: Bandwidth is an issue when rerouting a large amount of data in the SCADA network. We consider the alternate paths when calculating the bandwidth requirements. The shared links on the paths need to be of high enough bandwidth to allow the traffic from the sharing devices through them. It is not desirable to clog up a certain link by redirecting all packets through a certain path. It is important to ensure that there is at least one path from an IED to the MTU that has enough bandwidth on each link

so that the data collected by that IED can safely reach the destinations. Let $delData_i$ be the data to be delivered from IED i. $AssuredBandwidth_i$ assures the required bandwidth:

$$\forall_{i\in\mathbb{I}} AssuredBandwidth_i \;\leftrightarrow\; \exists_{p_{i,y}\in\mathbb{P}_i}\forall_{l\in p_{i,y}} bandwidth_l \geq \sum_i delData_i \tag{8}$$

At this point, we model the assured delivery of data from IED i to MTU. If it is assured that there is at least one SDN-enabled switch on the path from an IED to the MTU, and there are some assured alternate paths between the communicating parties, we conclude that data delivery is assured from that IED to the MTU. The bandwidth on all links on at least one of the paths must also be assured. We formalize this as follows:

$$\forall_{i\in\mathbb{I}} AssuredDelivery_i \;\rightarrow\; SDAltPath_{p_{i,y},p_{i,y'}} \wedge AssuredBandwidth_i \wedge AssuredAltPath_i \tag{9}$$

F. Virtualization and Isolation

SDN offers the capability of network virtualization, which can group intelligent devices into logically isolated virtual networks. Virtualization can also isolate different flows of data, which makes it easier to perform separate control on traffic with different interest [40]. Virtualization technologies, such as VLANs or VPNs also become feasible using SDN, as tedious and error-prone work of configuring of individual switches to create virtual networks is eliminated. SDN enables this process to be automated because an overlay network built on SDN can be reconfigured quickly according to software instructions [4].

VLANs can help to isolate different parts of a SCADA system. For example, the historian and the HMI machines in the control center may have more tendency to connect to the Internet. These groups of devices can be isolated using a VLAN from all other devices that need to be hidden from potential attack networks. Again, groups of vulnerable and critical devices may be required to be scanned using a system such as IDS [41]. The IEDs in the threat vectors are also good candidates to be in VLANs, as they can be secured through their private networks. VLANs can be configured to scrutinize all packets from these devices by passing them through such a detection technology. One other use of VLANs can be to isolate any known compromised devices from the core network, so that they cannot infect the other devices easily. SDN gives us the benefit of dynamically creating these VLANs as per the requirements of a smart grid SCADA network.

Let $Vlan_v$ be the v^{th} VLAN in the SCADA network, which is basically a set of IEDs. For simplicity, we assume that if a switch is connected to an IED that is in any VLAN, it is considered to be a VLAN enabling switch. If $VlanEnablingSwitch_s$ indicates whether switch s can enable a VLAN for one of its connected hosts, and S_v denotes the set of switches for VLAN v, the following holds:

$$\forall_v \forall_{i\in Vlan_v}\; \forall_{s\in S_v}\; VlanEnablingSwitch_s \;\rightarrow\; Link_{i,s} \tag{10}$$

The union of all the switches in all VLANs should be a subset of the set of all the switches: $\bigcup_v\; S_v \subset \mathbb{S}$. If a switch needs configuration for putting an IED under a VLAN, that is, if it is a VLAN enabling switch, it should be an SDN switch.

$$\forall_{s\in\mathbb{S}} VlanEnablingSwitch_s \;\rightarrow\; SwitchIsSDN_s \tag{11}$$

It is obvious that the number of available SDN switches is limited and might not be enough to create all the VLANs that we require. We define Q_v as the ratio for a VLAN as the number of SDN-enabled switches to the total number of switches in that VLAN.

$$Q_v = \frac{\sum_{s\in S_v}\; SwitchIsSDN_s}{|S_v|} \leq 1 \tag{12}$$

If *vlanExp* indicates the expected percentage of implementing the desired VLANs specified by users in the input file and V indicates the total number of VLANs, then,

$$\frac{\sum_v Q_v}{V} \geq vlanExp \tag{13}$$

G. Resources and Budget Constraints

It is important in an SDN environment that any communication is supervised by the SDN controller. This means that any packet from a source must traverse through at least one SDN switch on its way to the destination. Let $ESwitchIsSDN_s$ denote an already existing SDN switch s and $DSwitchIsSDN_s$ be the deployable new SDN switch. The following should hold about the already existing switches or routers and the newly deployable SDN-enabled switches.

$$SwitchIsSDN_s \rightarrow ESwitchIsSDN_s \vee DSwitchIsSDN_s \tag{14}$$

$$(ESwitchIsSDN_s \;\rightarrow\; \neg DSwitchIsSDN_s) \wedge (DSwitchIsSDN_s \;\rightarrow\; \neg ESwitchIsSDN_s) \tag{15}$$

Similarly, if $DLink_l$ denotes a newly deployed link that does not exist and need to be set up, then,

$$Link_l \;\rightarrow\; ELink_l \;\vee\; DLink_l \tag{16}$$

$$(ELink_l \;\rightarrow\; \neg DLink_l) \wedge (DLink_l \;\rightarrow\; \neg ELink_l) \tag{17}$$

The total budget, *TOT_AVAIL_BUDGET*, hence the number of available SDN-enabled switches and links are limited. It is not possible to replace a greater number of switches than the total available SDN switches. That is, the sum of all deployed SDN switches and links must be less than what we have in budget. This can be represented by the following constraint, given c_{Link_l} represents the deployment cost of new $DLink_l$ and c_{SDN} denotes the cost of each SDN switch:

$$(\sum_l DLink_l) \times c_{Link_l} + (\sum_s DSwitchIsSDN_s) \times c_{SDN} \leq TOT\_AVAIL\_BUDGET \tag{18}$$

IV. Prototype Implementation of SDNSynth and An Example Case Study

The main objective of our configuration synthesis problem is to synthesize the network topology for implementation of SDN by satisfying various requirements as well as the business constraints of a SCADA network in smart grids. Thus, the synthesis problem is formalized as the satisfaction of the conjunction of all the constraints specified in section III.

A. Target Variables in the Model

We implement our model by encoding the system configuration and the constraints into SMT logics [42]. In this encoding purpose, we use Z3, an efficient SMT solver [43]. We use boolean terms for encoding the boolean configuration parameters and decision variables like SDN switch deployments. The remaining parameters are modeled as integer or real terms.

The solver checks the verification constraints and provides a satisfiable (*sat*) result if all the constraints are satisfied. The *sat* result provides a *sat* instance, which represents the value assignments to the parameters of the model. According to our objective, we require the assignments to the following variable: the decision variable referring to whether a switch is SDN-enabled, $SwitchIsSDN_s$, *i.e.*, the placement of the SDN-enabled switches. The other target variable is $DLink_l$, which represents if link l should be deployed or not. The values of these parameters are printed in a text file (output file).

Although the solution using the Z3 solver may not always provide an optimal solution, it can provide solutions that are very close to optimal. In the evaluation results, we show the required number of SDN switches and links from the tightest possible constraints that provide *sat* results. In fact, when we start with a budget too low, it will provide *unsat* results. When we keep increasing the budget at little quantities at a time, at some point, it will provide a *sat* solution if all other constraints satisfy. This solution is very close to optimal. It is also possible to create a binary-search-based algorithm that can provide the best solution among many available ones. As the problem in this research is a combinatorially hard problem, an efficient solution is required for acceptability. Z3 is one of the most efficient solvers of formal models.

B. A Synthetic Pedagogical Case Study

Fig. 2(a) shows a small network of a 14-bus SCADA system. It consists of 26 IEDs, 13 RTUs, and 1 MTU. There are 18 traditional routers connecting these intelligent devices. Here we present three case studies, as demonstrated by SDNSynth. The input file consists of the network topology, the number of possible new links, the data generated by the IEDs, etc. It also includes 8 threat vectors where each is a set of IEDs, the expected percentage of threat mitigation (75%), average cost of new SDN switches ($2000), the available budget ($35,000), the desired VLANs specifications, the expected alternate path percentage for critical devices (70%), etc.

A Satisfiable Case Study: First, we determine the k-resiliency of the SCADA network, which means the network is resilient to less than any k IED failures [8]. In other words, the system is still observable if less than k IEDs fail to deliver their measurement data to the MTU. The system becomes unobservable if an attacker is capable of compromising equal to or more than k devices at a time. The value of k in our first network in Fig. 2(a) is 4, which means that it is 3-device resilient. In other words, the network can sustain up to 3 IED failures. If the attacker is capable of compromising up to 4 devices, this scenario yields 1 threat vector consisting of 4 devices; whereas, for a 5-device failure we get 8 different threat vectors. We use the 8-threat vector scenario for our first case study. Table II shows the input file used in this case.

TABLE II
THE INPUT TO THE CASE STUDY

```
# Number of IEDs (1-26), RTUs (27-39) and Routers (40-57) (1 MTU (58) is assumed)
26 13 18
# IED generated data rate (Gbps)
2 1 1 2 2 3 4 1 2 1 1 2 2 3 4 1 2 1 1 2 2 3 4 1 3 4
# Number of links in topology
58
# Communication links (among the IEDs, the RTUs, and the MTU)
# From - To - Bandwidth (Gbps)
1 40 5
2 40 5
27 40 10
3 41 10
28 41 10
4 42 5
5 42 5
6 42 5
29 42 10
7 43 5
.........
52 56 10
57 58 10
# Number of possible new links in the network topology according to budget
6
# Possible new links
# From - To - Bandwidth (Gbps)
44 55 10
45 47 10
47 57 10
.........
52 57 10
# Number of threat vectors for the current topology
8
# Threat vectors (all the IEDs in each individual set cannot be compromised at once)
13 14 15 16
11 12 24 25 26
3 4 5 6 8
12 14 17 18 25 26
17 18 20 21 22 23
11 12 20 21 22 23
1 2 11 12 25 26
1 2 3 4 9 10
# Expected threat vector mitigation percentage
75
# Avg cost of SDN switch ($)
2000
# Cost of new links ($)
900 1100 1150 850 1050 950
# Available budget ($)
35000
# Number of desired VLANs
3
# Desired VLAN sets
1 2 4 9
5 6
10 11 23
# Expected percentage of SDN-enabled switches to be in the desired VLANs
50
# Expected alternate path percentage for critical devices
75
```

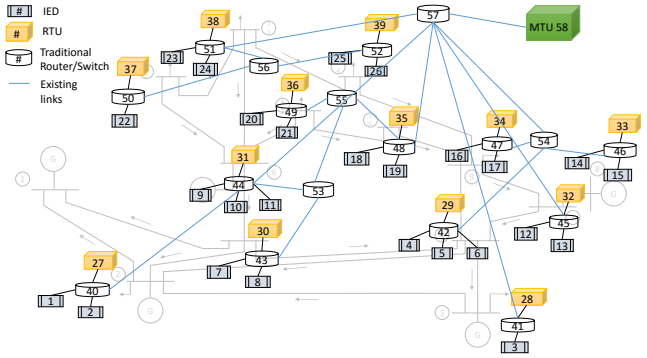
In this case, we have a budget of $35,000. The SDN-enabled switches should replace some of the traditional routers. The price of each new SDN-enabled switch and deployment cost of new links are provided in the input file. The amount of data generated by each of the IEDs is also provided, which is required to comply with the throughput of the links on its path to the MTU. For our 8 sets of threat vectors, the attacker has a capability of attacking at most 5 intelligent devices at a time. One of the threat vectors consists of IEDs 13, 14, 15 and 16, which means that if all of these IEDs are failed together due to an attack, then the system will be unobservable.

We would like to mitigate at least 75% of the threat

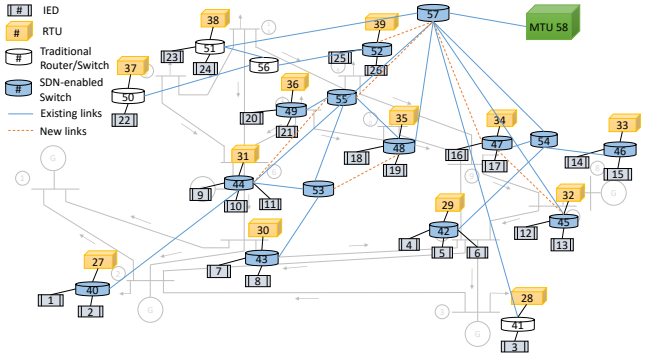
TABLE III
OUTPUT TO THE EXAMPLE

```
# We have a solution
# Routers required to be replaced with SDN-enabled ones
Router_ID Deployed?
40 True
41 False
42 True
...
54 False
...

# All existing and deployed links:
Source Dest Status New?
1 40 True False
2 40 True False
...
48 53 True True
...
Required time (ms): 5392.85
```



(a)



(b)

Fig. 2. (a) Traditional SCADA network in Smartgrid with legacy switches, (b) SDN network with Smartgrid after deployment of SDN-enabled switches.

vectors through the creation of VLANs and placement of SDN-enabled switches in this case study. We have a limited budget and 6 possible new links. Given all the constraints and requirements, SDNSynth generates multiple solutions for the provided input. The users may choose one solution that optimizes any particular parameter according to their requirements using a binary-search-based hill-climbing approach. One of the solutions, in this case study, tells the user to replace the routers 40, 42, 43, 44, and so on. Router 57 is also among the suggested replacement list, as it is almost on all possible alternate paths from an IED to the MTU. Fig. 2(b) demonstrates this scenario. Moreover, new links between switch 44 and 55, 45 and 47, 47 and 57, 52 and 57, and so on, are suggested. This solution provides a close-to-minimal number of switches and links required to be deployed to meet the requirements. If the threat mitigation requirement was less than 75%, SDNSynth might have provided a solution with even less number of SDN switches.

The proposed network topology ensures required bandwidths in each link from any IED to the MTU. It ensures alternate paths for critical IEDs according to the percentage of expected IEDs having alternate paths. These paths enable the SDN controller to reroute any critical data to and from the IEDs in the case of any link or device failure. It also ensures the forming of VLANs with critical IEDs by providing SDN-enabled switches. The controller can create VLANs utilizing the SDN-enabled switches with the IEDs that need to be secured from the compromised ones. It can also be used to isolate compromised devices. SDNSynth provides a software-defined network architecture (within the given capacity) that ensures that the percentage of IEDs in VLANs exceeds the expected specification of 75%.

Listing 1 presents an excerpt of the Z3 code generated by SDNSynth. It shows a few of the expressions that assert the constraints according to our resiliency model [1].

```
1. assert(=> AssuredBw_1 and((>= Bw_1_40 2)
          (>= Bw_40_44 2) (>= Bw_40_44 2)
          (>= Bw_44_55 2) (>= Bw_55_57 2) (>= Bw_57_58 2)))
2. assert(>= 35000
          (+ (* ((+ (SDNSwitchCount_1 SDNSwitchCount_2 ...
                    SDNSwitchCount_18))) 2000)
             (* ((+ (NewLinkCount_1 SDNSwitchCount_2 ...
                    SDNSwitchCount_6))) 1000)))
3. assert(=> AssuredAltPath_1
         and (AssuredMinAltPath_1 AssuredLinksOnAltPath_1))
4. assert(=> AssuredAltPath_2
         and (AssuredMinAltPath_2 AssuredLinksOnAltPath_2))
...
```

Listing 1. Sample Code

A Case with Increased Attacker Capability: In this case, we envision an attacker with more attacking capability, which means the number of threat vectors will increase proportionally. For 6 IED failures, the number of threat vectors increases to 42, where each vector can consist of as many as 6 IEDs. We keep the budget same and verify the model. With these constraints and requirements, SDNSynth generates an unsatisfiable result. This means that with the existing budget and the attacker capability, deployment of SDN-enabled switches may not be sufficient to repel the attacks.

Next, we increase the budget to $50,000. The number of possible links is also increased. This time SDNSynth provides a solution that tells the user to replace the routers 40, 41, 42, 43, 44, and so on. Routers 56 is also among the suggested replacement list. New links between switch 44 and 55; 47 and 54; 48 and 53; 55 and 57; 52 and 57; etc. are suggested, which makes the system more resilient by providing alternate paths for critical IEDs.

A Case with a Different SCADA Topology: Here we use a different SCADA topology than the one in Fig. 2(a) for a 14-bus test system. We generate the topology using a different 14-bus connectivity (lines) for the buses. The links between router 44 and 53; 48 and 55; and 42 and 54 are not available

[1] https://github.com/jakaria42/SDN_Topo_Synth.git

in this network. Instead, there is a link between 42 and 57. For this network, we get a 2-device resiliency, which means the network is vulnerable to as low as 3-IED failure. We consider an attacker to compromise at most 4 IEDs together. We get a threat vector count of 3 for this. From the results provided by SDNSynth, we observe that at least 12 SDN-enabled switches and all the 4 available new links are required to acquire a satisfiable solution. Also, we need a budget of at least $28,000.

V. Evaluation

In this section, we present the evaluation results showing the relationships between different network deployment parameters, such as the number of newly deployed SDN-enabled switches and links, available budget, threat vector mitigation requirements, etc. We also present the scalability of the proposed framework with respect to the SCADA systems. Lastly, we present a network simulation along with its results that depict the effectiveness of out proposed methodology. To the best of our knowledge, there is no other work that solve the same deployment problem with the resiliency of SCADA in mind. Hence, we do not present any comparison of performance with any existing technique.

A. Methodology

To evaluate SDNSynth, we run experiments on different SCADA network topology for different bus sizes in smart grids. We created synthetic networks based on the line connectivity of the buses and power flow and line measurements. We ran the program on a machine equipped with Windows 10 OS, an Intel Core i7 processor and 16 GB memory. We evaluate the scalability of the verification model by the analysis of the time requirements for executing the model in different scenarios. It is worth mentioning that the number of intelligent devices (IEDs and RTUs), as well as the routers and links are not fixed for a specific bus size for a SCADA system. However, their numbers are generally proportional to the bus sizes. We generate the SCADA networks based on different sizes of IEEE test systems, *i.e.*, 14-bus, 30-bus, 57-bus, and 118-bus [44]. The communication paths from an IED to the MTU are formed randomly considering several routers/switches.

B. Relationships of Deployment Parameters

The intention behind Fig. 3 and Fig. 4 is to show that the parameters used in our model relate to each other as expected. This entails the correctness of the model. In this analysis, the resultant numbers of SDN-enabled switches and links are the minimum numbers of switches and links possible with the tightest possible constraints.

Fig. 3(a) shows the relationship between the number of threat vectors and the minimum number of deployed SDN-enabled switches. We observe from the graph that the number of SDN switches increases with the threat vectors. For 60% threat mitigation requirement, the number of IEDs in the threat vectors that are required to be resilient, is less than that of 90%. As a result, we need fewer SDN-enabled switches for the 60% case than the 90% case. For example, when we need to mitigate only 60% of 42 threat vectors, 20 new switches are required, whereas, 31 switches are required to mitigate 90%. Fig. 3(b) shows similar results for newly deployed links. In Fig. 3(c), we present the minimum required budget for different numbers of threat vectors in the SCADA system of a 57-bus grid. More budget is required to mitigate more threats for a certain number of threat vectors. A requirement of 90% threat mitigation requires more budget than the 60% case.

In Fig. 3(d), we show the results for different existing network topologies of a 30-bus SCADA system. We change the average degree of the existing routers, where the degree of a router refers to the number of links attached to it. We observe that the number of required new switches decreases as the degree increases. For higher degrees of routers, there are already higher numbers of alternate paths. Hence, it requires less number of SDN-enabled switches to be resilient to threats.

Fig. 4(a) shows the percentage of threat vectors that can be mitigated with different amounts of budgets. We show two cases for 42 and 431 threat vectors for the same 57-bus SCADA network. As anticipated, a greater fraction of threat vectors can be mitigated with more budget by deploying enough new SDN-enabled switches and links. In Fig. 4(b), we present the percentage of VLAN creation possibility of the desired total VLANs. For a certain budget, when the total number of desired VLANs is 25, the percentage of fulfillment is less than that of expected VLAN count of 10. In Fig. 4(c), we demonstrate the number of deployed SDN-enabled switches, as well as the number of newly deployed links, with respect to the total available budget for 14 and 30 bus systems, while other parameters such as threat mitigation requirement etc. remain the same. As the available budget increases, the number of newly deployed SDN switches and links also increases slowly. The resource constraints are responsible for this increase, as SDNSynth tries to find a solution utilizing the available budget. New links ensure the proper rerouting of data for critical devices. Proper positioning of the SDN switches allows rerouting, as well as creation of VLANs with a particular group of IEDs.

We also vary the total number of devices in the 30-bus SCADA system, and observe the number of deployed SDN-enabled switches in Fig. 4(d). We do this for 8 and 42 threat vectors while keeping the threat mitigation requirement constant at 60%. The number of SDN switches increases very slowly for different sizes of the network, if the number of threat vectors, budget, threat mitigation requirement, and other parameters remain the same.

C. Scalability

We evaluate the scalability of SDNSynth by analyzing the time required to synthesize the network topology by varying the problem size and other parameters. The synthesis time includes the model generation time and the constraint verification time. All the times are measured for producing an efficient result by the SMT solver. While there are other greedy heuristic-based approaches available for incremental deployment, they do not address the same problem as ours. Moreover, greedy solutions are prone to get stuck in a local minima or maxima, while SDNSynth provides efficient results.

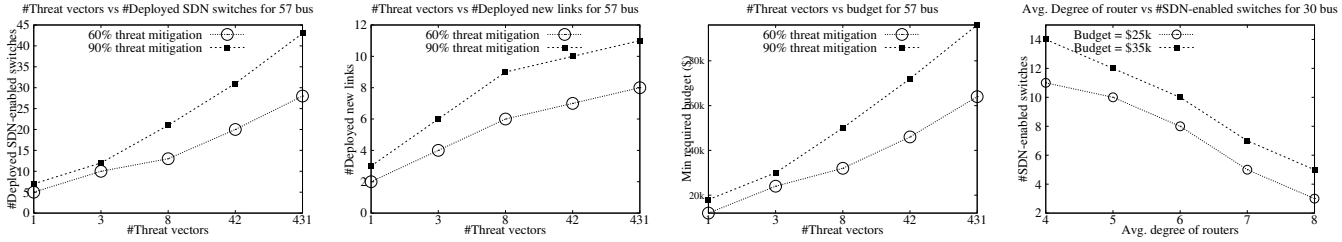


Fig. 3. (a) Number of required SDN switches w.r.t. number of threat vectors, (b) number of required new links w.r.t. threat vectors, (c) required budget w.r.t. threat vectors, (d) number of required SDN switches w.r.t. avg. router degree.

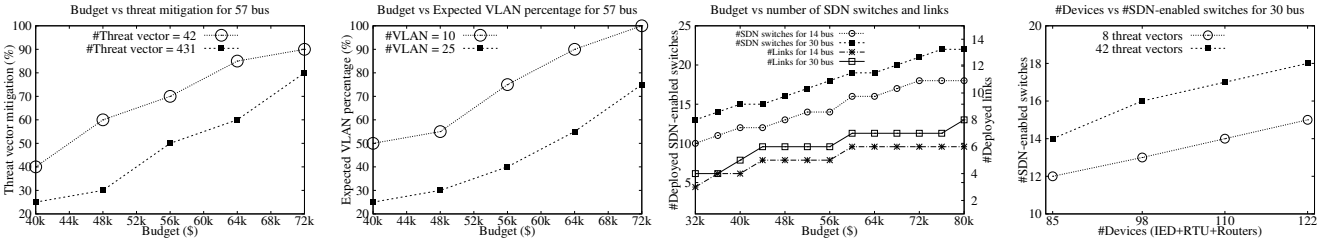


Fig. 4. (a) The percentage of threat vector mitigation w.r.t. budget, (b) the expected VLAN percentage w.r.t. budget, (c) number of SDN switches w.r.t. budget, and (d) number of SDN switches w.r.t. number of devices in SCADA.

Impact of Bus Size: The model synthesis time with respect to the varying bus size is shown in Fig. 5(a). Two scenarios, one for satisfiable results and another for unsatisfiable results, are presented in the graph for 8 threat vectors. We observe that the required time increases in somewhere between linear and quadratic orders with the increment of bus size. The execution time differs for *sat* and *unsat* results for a specific bus size. The *unsat* results usually take more time than *sat* ones. As the bus size increases, the number of constraints and requirements increase rapidly. For this reason, we observe such timing (almost quadratic) for obtaining a result.

Impact of SCADA Network Size: We also observe the model synthesis time by varying the number of devices (IEDs, RTUs, and routers) in a 30 bus network, while keeping the number of threat vectors constant in Fig. 5(b). We observe the time for scenarios with 8 and 42 threat vectors by increasing the number of devices from 58 to 101. We generate different numbers of devices by changing the average number of measurements per device in the same bus system. It was observed that the time increases almost linearly with the increase of the number of devices. As the number of available devices increases, the problem size increases in terms of the number of possible deployment positions of SDN-enabled switches. Verification of more constraints is required as the model size increases; hence, more time is required to reach a solution.

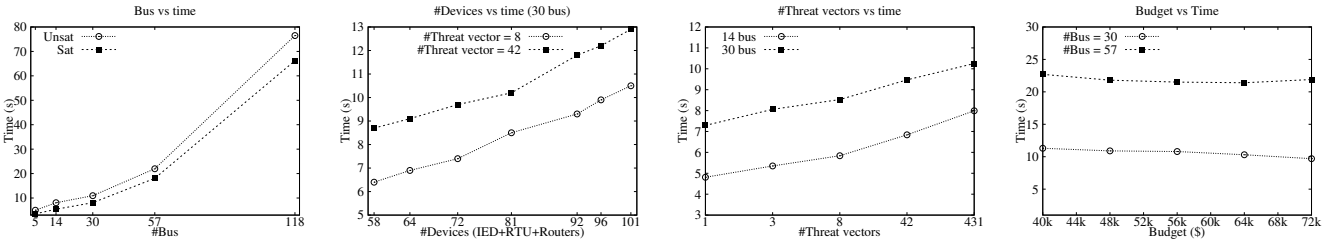
Impact of Threat Vectors: In Fig. 5(c), the time taken by the Z3 solver was measured for a varying number of threat vectors, where the number of threat vectors is almost quadratically proportional to the capability of an attacker. For a certain bus size, as the number of threat vectors increases, the number of Z3 clauses to verify also gradually increases. For a certain number of threat vectors, the difference is not too much between 14 and 30 bus systems. It is worth mentioning that the x-axes for Fig. 3(a) through Fig. 3(c) and Fig. 5(c) are not linear and displayed without scale.

Impact of Budget: Fig. 5(d) shows the impact of budget on the network synthesis time. We can observe that if the bus and SCADA size, threat vectors, as well as all other requirements are kept constant, the tool requires almost similar times for synthesizing the network. The network size is larger for the 57 bus than the 30 bus. As a result, there are more constraints to solve, hence it takes more time to provide a satisfiable result.

D. Experiments on Mininet-based Virtual SCADA Testbeds

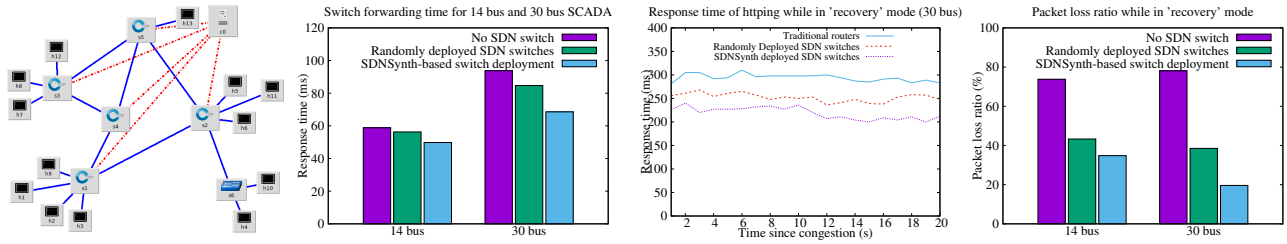
We simulate the SCADA networks of different IEEE test bus systems with both legacy and SDN-enabled switches. We use Mininet version 2.2.2 [45] in VirtualBox 5.2.0. Mininet is a open-source network emulator virtual machine that runs a real kernel, switch and application code. It creates a network of virtual hosts, switches, controllers, and links. Mininet hosts run standard Linux network software. It creates realistic virtual networks consisting of both legacy switches and SDN-enabled switches supporting OpenFlow. We configure the network of the Mininet VM to contain a host-only network interface, which lets us SSH into Mininet from the host system. We run the 'Xming' server on a Windows host machine to enable X11 forwarding, which facilitates the launch of network monitors such as Wireshark, xterm, etc. We use a python application called 'Miniedit' within Mininet which lets the user to use a graphical interface to create networks. This application can also export the topology to customize the behavior of the network, such as using remote controllers running at a certain IP address and port.

Although Mininet has its default OpenFlow reference controller, we use a remote Floodlight controller [46]. Floodlight is a Java-based OpenFlow controller for enterprise networks distributed under Apache license. We use the latest master version of Floodlight with Java 8. It utilizes the collaboration between two modules named *LinkDiscoverManager* and *TopologyService* to be aware of the topology, and automatically learns about networks with loops and alternate paths in



(a) (b) (c) (d)

Fig. 5. The model synthesis time w.r.t. (a) bus size, (b) number of devices, (c) number of threat vectors, and (d) available budget.



(a) (b) (c) (d)

Fig. 6. (a) Sample 5 bus SCADA network in Miniedit environment of Mininet with both SDN-enabled and legacy switches, (b) Node forwarding time in terms of avg. response time of Ping commands, (c) Comparison of response time of 'httping' while recovering during congestion for different types of networks (30 bus), and (d) Packet loss ratio while recovering from congestion for different types of networks.

the network. Floodlight specifies a protocol which is utilized by a remote controller to modify the behavior of networking devices, such as Open vSwitches, through a well-defined 'forwarding instruction set'. We run Floodlight on a separate VM and connect it to Mininet through the host-only network.

Simulation Environment Specifications: To host the virtual machines, we use a computer with Windows 10 64-bit operating system with Intel(R) Core(TM) i5 670 3.47 GHz processor, and 12 GB of RAM. The Oracle VirtualBox VM manager manages the Mininet Emulator on a Linux operating system of Ubuntu 14.04.4-server-i386 32 bit, with 1 GB of memory. It also manages the Floodlight VM on Linux operating system Ubuntu 14.04.1 64 bit with 1 GB of memory.

Experiment Setup and Results: In our simulation, we specifically present three experiments to show the effectiveness of SDN in networks such as SCADA, as well as the role of SDNSynth in creating a resilient network infrastructure. In this experiment, we deploy three network simulation scenarios in Mininet: one without any SDN-enabled switches, one with randomly deployed SDN-enabled switches, and lastly SDNSynth suggested deployment of SDN switches and links. For the randomly deployed SDN topology scenario in Fig. 6(b) and 6(d), we took the average resultant metric from 30 randomly created topologies. For Fig. 6(c), we plot the median of the 30 curves generated from the random runs.

1) Switch Forwarding Speed: We compare the speed of forwarding switches/routers in network scenarios for 14 bus and 30 bus systems. To do this, we compute the response time of Ping command between two hosts in the network, which are assumed to be an IED and the MTU. Fig. 6(a) shows an example network for a 5 bus system in Miniedit of the Mininet VM, where host 1 is an IED which is an element of a threat vector for the network, while host 13 is the MTU. The response time is mainly composed of link delay, packet forwarding time, and routing table or flow table querying and matching time. We set the link bandwidth to 10 Mbps, link delay to 15 ms, and the packet loss rate in the links to 0.

Fig. 6(b) shows the observation for average response time of the Ping packets for each of the three scenarios. In 14 bus system SCADA network, the response time is almost the same for all scenarios, whereas in 30 bus system, the difference is much more observable. As the network size increases, the benefits of the software-defined network become more apparent. In both the cases, SDNSynth suggested topology performs better than the topology where SDN switches are randomly deployed.

In traditional network, with the expansion of the network size, the size of the routing table also increases considerably. The routing table is queried for each incoming packet after it has been established. In SDN-enabled network, only the first packet in a flow needs extra time to make a round trip from the controller, then the flow-tables are updated in the switch, which are much simpler and smaller in size for similar network topology. This causes the forwarding speed of Open vSwitches in SDN networks to be much faster. Moreover, SDNSynth proposes the strategic deployment of switches to ensure that most of the packets are dispatched through the SDN-enabled Open vSwitches.

2) Communication Time While Routing Recovery for Resilient Networks: To calculate the communication time, we consider both the latency of the network and the webserver. We use a tool named 'httping' [47] that can measure the time to connect to a webserver, send a request and retrieve the reply. We set up a simple http server on a host in the Mininet network that acts as an MTU. The hosts acting as field devices such as IEDs use 'httping' to communicate with the MTU.

In this experiment, we show the benefits of priority-based routing in SDN-enabled switches. We create three different networks of similar topology with SDNSynth suggested Open vSwitches, randomly deployed Open vSwitches, and only traditional routers. We consider some hosts (IEDs) in the

networks having higher priority than others, as they are part of threat vectors, as discussed in Section III-C. All the hosts acting as IEDs are active and perform continuous communications with the MTU. When the network is established, we manually create some link failures to simulate a congestion. We take some direct links of the high priority IEDs down, so that the traffic from them are rerouted through alternate paths. The bandwidth of the links is set to 10 Mbps and the delay is set to 30 ms. In the Open vSwitches, we manually set high priorities for the flows from IEDs that have higher priority by executing commands like the following from the CLI:

```
sh ovs-ofctl add-flow s1 priority=1500,dl_type=0x800,
nw_src=10.0.0.1/24,nw_dst=10.0.0.1/24,actions=normal
```

Fig. 6(c) shows the communication times between a high priory host (IED) and the server host (MTU) recorded by 'httping' when the network is in 'recovery' mode. The graphs show the communication times for the first 20 seconds since the link failures occurred. Fig. 6(c) shows the results for a network for 30 bus with 90 switches/routers. The results show that the systematic deployment of SDN-enabled switches offers much faster recovery for high priority end devices in case of congestion in the network. In randomly deployed SDN networks, the response time for the httping packets is higher than a systematically deployed network. In traditional networks, the switches do not have the option of dynamically setting up priories for certain hosts in the network. As a result, traditional networks show much higher communication times during congestion situations. It can be observed from the graphs that the benefit of SDNSynth deployment is much more apparent in the larger network of 30 bus. The larger the network is, the better SDNSynth suggested networks do than the other two deployments.

3) Packet Loss Ratio For Resilient Networks: We use the same experimental set up as described in Section V-D2, except that we increase the link delay in the networks to 50 ms, which causes a considerable amount of packet loss in traditional networks. We calculate the packet loss ratio for a high priority host (IED) in the case of a congestion created by a manual link failure. It is calculated as the ratio of lost packets to the total transmitted packes. Figure 6(d) shows the average packet loss ratio for the first 100 pings after the congestion is created. We use the 'fping' tool to control the time-out period and measure the loss ratio [48]. It can be observed that the packet loss ratio for SDNSynth suggested network is much lower than the other two scenarios. As SDNSynth deploys Open vSwitches in such a manner that most of the packets should pass through the Open vSwitches, the number of lost packets is decreased. These switches prioritize the flows from higher priority hosts when forwarding. It can be observed that the loss ratio is much less in the larger network of 30 bus suggested by SDNSynth.

VI. Conclusion

SDNSynth provides a tool for synthesizing a resilient SDN network topology in smart grid SCADA systems. We address practical challenges in the deployment of SDN-enabled switches while not exceeding available budget. Our goal is to keep the SCADA network observable in terms of state estimation by means of sufficient measurement delivery to the control center. We protect smart devices such as IEDs, with the use of SDN-enabled switches, which allow fast rerouting, prioritization of packet flows, novel application-based routing, etc. The availability of new SDN switches for replacement of traditional ones is limited. The technique successfully generates a solution depicting the SDN switch and new link placements, while satisfying the resiliency requirements and budget constraints. We evaluate the tool by evaluating the scalability and satisfactory relationship between different deployment parameters. A simulation of SCADA networks in Mininet depicts that SDNSynth generated topology shows faster command response time than random selection of switches for SDN upgrade. In the future, we would like to experiment on a real-world SDN network and demonstrate the benefits of our framework. We plan to use the Global Environment for Network Innovations (GENI) testbed to build a hybrid network of SDN and lagacy routers [49]. The GENI infrastructure allows the use of Floodlight controller in conjunction with Open vSwitches, where we plan to compare different network scenarios to demonstrate the effectiveness of SDNSynth.

References

[1] V. C. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. P. Hancke, "Smart grid technologies: Communication technologies and standards," *IEEE transactions on Industrial informatics*, vol. 7, no. 4, pp. 529–539, 2011.

[2] G. R. Clarke, D. Reynders, and E. Wright, *Practical modern SCADA protocols: DNP3, 60870.5 and related systems*. Newnes, 2004.

[3] A. Cahn, J. Hoyos, M. Hulse, and E. Keller, "Software-defined energy communication networks: From substation automation to future smart grids," in *Smart Grid Communications (SmartGridComm), 2013 IEEE International Conference on*. IEEE, 2013, pp. 558–563.

[4] J. Zhang, B.-C. Seet, T.-T. Lie, and C. H. Foh, "Opportunities for software-defined networking in smart grid," in *Information, Communications and Signal Processing (ICICS) 2013 9th International Conference on*. IEEE, 2013, pp. 1–5.

[5] L. Ren, Y. Qin, B. Wang, P. Zhang, P. B. Luh, and R. Jin, "Enabling resilient microgrid through programmable network," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 2826–2836, 2017.

[6] D. K. Hong, Y. Ma, S. Banerjee, and Z. M. Mao, "Incremental deployment of sdn in hybrid enterprise and isp networks," in *Proceedings of the Symposium on SDN Research*. ACM, 2016, p. 1.

[7] H. Xu, X.-Y. Li, L. Huang, H. Deng, H. Huang, and H. Wang, "Incremental deployment and throughput maximization routing for a hybrid sdn," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1861–1875, 2017.

[8] M. A. Rahman, A. Jakaria, and E. Al-Shaer, "Formal analysis for dependable supervisory control and data acquisition in smart grids," in *Dependable Systems and Networks (DSN), 2016 46th Annual IEEE/IFIP International Conference on*. IEEE, 2016, pp. 263–274.

[9] A. Jakaria, M. A. Rahman, and C. J. Fungt, "Automated synthesis of nfv topology: A security requirement-oriented design," in *2017 13th International Conference on Network and Service Management (CNSM)*. IEEE, 2017, pp. 1–5.

[10] K. Stouffer and J. Falco, *Guide to supervisory control and data acquisition (SCADA) and industrial control systems security*. National institute of standards and technology, 2006.

[11] E. Al-Shaer, M. A. Rahman *et al.*, "Security and resiliency analytics for smart grids," *Advances in Information Security*, 2016.

[12] K. Tomsovic, D. E. Bakken, V. Venkatasubramanian, and A. Bose, "Designing the next generation of real-time control, communication, and computations for large power systems," *Proceedings of the IEEE*, vol. 93, no. 5, pp. 965–979, 2005.

[13] V. M. Igure, S. A. Laughter, and R. D. Williams, "Security issues in SCADA networks," *Comp & Security*, vol. 25, no. 7, pp. 498–506, 2006.

[14] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[15] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[16] N. Dorsch, F. Kurtz, H. Georg, C. Hägerling, and C. Wietfeld, "Software-defined networking for smart grid communications: Applications, challenges and advantages," in *SmartGridComm, 2014 IEEE International Conference on*. IEEE, 2014, pp. 422–427.

[17] X. Dong, H. Lin, R. Tan, R. K. Iyer, and Z. Kalbarczyk, "Software-defined networking for smart grid resilience: Opportunities and challenges," in *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*. ACM, 2015, pp. 61–68.

[18] K. C. Budka, J. G. Deshpande, T. L. Doumi, M. Madden, and T. Mew, "Communication network architecture and design principles for smart grids," *Bell Labs Tech Journal*, vol. 15, no. 2, pp. 205–227, 2010.

[19] J. Kim, F. Filali, and Y.-B. Ko, "Trends and potentials of the smart grid infrastructure: from ict sub-system to sdn-enabled smart grid architecture," *Applied Sciences*, vol. 5, no. 4, pp. 706–727, 2015.

[20] D. A. Chekired, L. Khoukhi, and H. T. Mouftah, "Decentralized cloud-sdn architecture in smart grid: A dynamic pricing model," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 3, pp. 1220–1231, 2018.

[21] D. Jin, Z. Li, C. Hannon, C. Chen, J. Wang, M. Shahidehpour, and C. W. Lee, "Toward a cyber resilient and secure microgrid using software-defined networking," *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2494–2504, 2017.

[22] L. Ren, Y. Qin, Y. Li, P. Zhang, B. Wang, P. B. Luh, S. Han, T. Orekan, and T. Gong, "Enabling resilient distributed power sharing in networked microgrids through software defined networking," *Applied Energy*, vol. 210, pp. 1251–1265, 2018.

[23] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: an intellectual history of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, 2014.

[24] E. G. da Silva, L. A. D. Knob, J. A. Wickboldt, L. P. Gaspary, L. Z. Granville, and A. Schaeffer-Filho, "Capitalizing on sdn-based scada systems: An anti-eavesdropping case-study," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015, pp. 165–173.

[25] E. G. da Silva, A. S. da Silva, J. A. Wickboldt, P. Smith, L. Z. Granville, and A. Schaeffer-Filho, "A one-class nids for sdn-based scada systems," in *2016 IEEE 40th Annual Comp Software and Applications Conf (COMPSAC)*, vol. 1. IEEE, 2016, pp. 303–312.

[26] Y. Guo, Z. Wang, X. Yin, X. Shi, J. Wu, and H. Zhang, "Incremental deployment for traffic engineering in hybrid sdn network," in *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2015, pp. 1–8.

[27] D. Levin, M. Canini, S. Schmid, F. Schaffert, A. Feldmann *et al.*, "Panopticon: Reaping the benefits of incremental sdn deployment in enterprise networks." in *USENIX Annual Tech Conf*, 2014, pp. 333–345.

[28] D. Levin, M. Canini, S. Schmid, and A. Feldmann, "Incremental sdn deployment in enterprise networks," in *ACM SIGCOMM Computer Comm Review*, vol. 43, no. 4. ACM, 2013, pp. 473–474.

[29] P. Lin, J. Hart, U. Krishnaswamy, T. Murakami, M. Kobayashi, A. Al-Shabibi, K.-C. Wang, and J. Bi, "Seamless interworking of sdn and ip," in *ACM SIGCOMM computer communication review*, vol. 43, no. 4. ACM, 2013, pp. 475–476.

[30] S. Agarwal, M. Kodialam, and T. Lakshman, "Traffic engineering in software defined networks," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 2211–2219.

[31] S. Das, G. Parulkar, and N. McKeown, "Why openflow/sdn can succeed where gmpls failed," in *European Conference and Exhibition on Optical Communication*. Optical Society of America, 2012, pp. Tu–1.

[32] K. Poularakis, G. Iosifidis, G. Smaragdakis, and L. Tassiulas, "One step at a time: Optimizing sdn upgrades in isp networks," in *IEEE INFOCOM 2017-IEEE Conf on Computer Comm*. IEEE, 2017, pp. 1–9.

[33] C. Chen, S. Duan, T. Cai, B. Liu, and G. Hu, "Smart energy management system for optimal microgrid economic operation," *IET renewable power generation*, vol. 5, no. 3, pp. 258–267, 2011.

[34] A. Monticelli, "Electric power system state estimation," *Proceedings of the IEEE*, vol. 88, no. 2, pp. 262–282, 2000.

[35] M. A. Rahman, E. Al Shaer, and R. G. Kavasseri, "Security threat analytics and countermeasure synthesis for power system state estimation," in *2014 44th Annual IEEE/IFIP International Conf on Dependable Sys and Networks*. IEEE, 2014, pp. 156–167.

[36] A. Gomez-Exposito and A. Abur, *Power system state estimation: theory and implementation*. CRC press, 2004.

[37] M. A. Rahman, E. Al-Shaer, and M. A. Rahman, "A formal model for verifying stealthy attacks on state estimation in power grids," in *Smart Grid Communications (SmartGridComm), 2013 IEEE International Conference on*. IEEE, 2013, pp. 414–419.

[38] A. J. Wood and B. F. Wollenberg, *Power Generation, Operation, and Control, 2nd Edition*. Wiley, 1996.

[39] A. Goodney, S. Kumar, A. Ravi, and Y. H. Cho, "Efficient pmu networking with software defined networks," in *Smart Grid Communications (SmartGridComm), 2013 IEEE International Conference on*. IEEE, 2013, pp. 378–383.

[40] S. Gutz, A. Story, C. Schlesinger, and N. Foster, "Splendid isolation: A slice abstraction for software-defined networks," in *Proceedings of the first workshop on Hot topics in SDN*. ACM, 2012, pp. 79–84.

[41] "Applying VLAN insertion in ICS/SCADA," https://www.paloaltonetworks.com/resources/whitepapers/applying-vlan-insertion-in-ics-scada.

[42] L. de Moura and N. Bjørner, "Satisfiability modulo theories: An appetizer," in *Brazilian Symposium on Formal Methods*, 2009.

[43] L. De Moura and N. Bjørner, "Z3: An efficient SMT solver," *Tools and Algorithms for the Construction and Analysis of Sys*, pp. 337–340, 2008.

[44] "Power systems test case archive," http://www.ee.washington.edu/research/pstca/.

[45] "Mininet: An instant virtual network on your laptop (or other PC)," http://mininet.org/.

[46] "Floodlight OpenFlow Controller - Project Floodligh," http://www.projectfloodlight.org/floodlight/.

[47] "Httping," https://www.vanheusden.com/httping/.

[48] "fping," http://fping.sourceforge.net/man/.

[49] "Geni," https://groups.geni.net/geni/wiki/GENIExperimenter/Tutorials/OpenFlowOVS-Floodlight.

A H M Jakaria obtained his PhD degree in Computer Science at Tennessee Tech University, USA in May 2020. He is currently working as an Engineer/-Scientist III at the Electric Power Research Institute (EPRI). He received his BS in Computer Science and Engineering from Bangladesh University of Engineering and Technology, Dhaka in 2009. He obtained his MS in Computer Science from Tennessee Tech in 2019. Jakaria's primary research area includes information and network security for smart cyber-physical systems using NFV and SDN. He is also interested in resiliency issues in UAV and IoT networks. He focuses on the formal modeling of the problems and solving them efficiently for automated synthesis of network topology and management strategies.

Mohammad Ashiqur Rahman is an Assistant Professor in the Department of Electrical and Computer Engineering at Florida International University, USA. He received the BS and MS degrees in computer science and engineering from Bangladesh University of Engineering and Technology, Dhaka, in 2004 and 2007, respectively, and obtained the PhD degree in computing and information systems from the University of North Carolina at Charlotte in 2015. Rahman's research area covers a wide area of computer networks that includes computer and information security in both cyber and cyber-physical systems. His research interest includes formal security analysis, risk assessment and security hardening, secure and dependable resource management, and distributed computing.

Aniruddha Gokhale is a Professor at the Vanderbilt University. He received his BE from University of Pune, India, MS degree from Arizona State University, and PhD degree in computer science from the Washington University, USA. His research focuses primarily on solving systems-level challenges by designing and implementing innovative algorithmic solutions incorporating elegant software engineering principles, such as design patterns, domain-specific modeling and generative programming. Specifically, he is interested in solving systems problems involving a variety of quality of service and data consistency issues through effective resource management, particularly in Cloud computing, Cyber Physical Systems, and Internet of Things.