Decentralized Intrusion Prevention (DIP) against Co-ordinated Cyberattacks on Distribution Automation Systems

Jennifer Appiah-Kubi, *Student Member, IEEE* Chen-Ching Liu, *Life Fellow, IEEE*

***Abstract*—Integration of Information and Communications Technology (ICT) into the distribution system makes today's power grid more remotely monitored and controlled than it has been. The fast increasing connectivity, however, also implies that the distribution grid today, or smart grid, is more vulnerable. Thus, research into intrusion/anomaly detection systems at the distribution level is in critical need. Current research on Intrusion Detection Systems for the power grid has been focused primarily on cyber security at the Supervisory Control And Data Acquisition, and single node levels with little attention on coordinated cyberattack at multiple nodes. A holistic approach toward system-wide cyber security for distribution systems is yet to be developed. This paper presents a novel approach toward intrusion prevention, using a multi-agent system, at the distribution system level. Simulations of the method have been performed on the IEEE 13-Node Test Feeder, and the results compared to those obtained from existing methods. The results have validated the performance of the proposed method for protection against cyber intrusions at the distribution system level.**

***Index Terms*—Cyber-physical system security, smart grid, distribution systems, intrusion detection, anomaly detection, multiagent system.**

Nomenclature

S	Set of likely attacks
F	Threshold for determining Denial of Service (DoS) attacks
P	Maximum allowed login attempts
ω_x	Weight assigned to an attack x
x_t	Threshold for determining attack x
x_r	Maximum recorded normal event related to attack x
ν	Attack potential
κ	Criticality index
ψ	Software vulnerability index
w_i	Weight assigned to an index i
ρ	Judgement value calculated in the second phase
T	Time period during which selected nodes stay in protective mode
n	Percentage reduction in thresholds
t	Maximum time allowed between two attacks, after which all nodes enter protective mode
$\mathcal{G}$	Graph model of distribution network
$\mathcal{V}$	Set of nodes of graph $\mathcal{G}$
$\mathcal{E}$	Set of edges of graph $\mathcal{G}$
k	An arbitrary graph state in agent communication
m	Number of nodes in graph $\mathcal{G}$
$\mathbf{A}(k)$	Adjacency matrix in state k of graph $\mathcal{G}$
$\mathbf{\Delta}(\mathbf{k})$	Degree matrix in state k of graph $\mathcal{G}$
$p^i(k)$	Vector in the memory of agent i for storing the values of its neighbors in the kth state
$u^i(k)$	Vector in the memory of agent i for storing the IDs of its neighbors in the kth state
$x_i(k)$	Selected value of agent i in the kth state of the third phase
$z_i(k)$	ID of the selected value of agent i in the kth state of the third phase
$\sigma_i(k)$	Message sent by agent i in the kth state of the third phase
D	Diameter of the network graph
$n_c(k)$	Number of communicating agents in state k

I. State-of-the-art

TRADITIONAL distribution grids have seen significant improvements that have made the grid more automated and remotely controllable. However, as a result, the grid is more vulnerable to cyberattacks. In December 2015, a series of cyberattacks were launched on six distribution utilities in the Ukrainian grid, causing an outage [1]. Research on distribution system cybersecurity is therefore critical.

Distribution Automation System (DAS) is an important part of the distribution system and may encompass Supervisory Control And Data Acquisition (SCADA). In the following subsections, a literature survey on cybersecurity for DAS and SCADA is provided, as well as for cybersecurity against coordinated cyberattacks, and solutions using multi-agent systems (MASs).

A. Supervisory Control And Data Acquisition (SCADA)

SCADA systems have evolved from traditional segregated networks to internet-connected remotely accessible systems. This, together with the use of legacy operating systems and protocols that were built with little to no consideration for cybersecurity, has significantly increased the attack surface for SCADA systems. While some generic cybersecurity measures, such as encryption, may be implemented, they are done within the limitations imposed by SCADA devices and networks, such as low computational capabilities, real time requirements and low data rate [2].

To study the cybersecurity issues of SCADA, several frameworks have been proposed. These include the integration framework [3], which presents a two-tiered architecture for

J. Appiah-Kubi is with the Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, 24061 USA. e-mail: jennifera@vt.edu.

C.-C. Liu is with the Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, 24061 USA. e-mail: ccliu@vt.edu.

incorporating simulation models into industrial systems. By incorporating real data sets into simulation models, operators are better able to understand the dynamics of the system and impact of cyberattacks. In reference [4], a three-level vulnerability assessment framework is proposed, that quantifies the vulnerability of the SCADA system at the access point, scenario and system-wide levels. Vulnerability assessment metrics are derived at each level. By using Bayesian attack graphs and a modified mean time to compromise (MTTC) parameter, the method of [5] determines the probabilities and outcomes for different attack paths on a SCADA network.

Some authors have proposed intrusion detection systems (IDSs). Reference [6] proposes a SCADA-specific IDS which combines access control white list with a protocol-based white list and a behavior-based rule set. Other IDSs are anomaly-based [7], and signature-based [8]. In addition, certain authors have suggested the use of honeypots to study attack behaviors [2].

B. Distribution Automation System (DAS)

DAS integrates communication with digital controls, switching devices, etc. to provide automated functionalities. Key Distribution Automation (DA) applications are outage management, feeder restoration, Volt-Var management, DER management, and condition monitoring. Some devices, e.g., smart reclosers, used in DA may be autonomous and not controlled from a remote location. On the other hand, remote control of field devices is also implemented. At the operations center, data is acquired by a SCADA system for processing and analysis. DA may also be integrated with applications such as Outage Management System (OMS), Distribution Management System (DMS) and Advanced Metering Infrastructure (AMI) [9].

The vulnerabilities of DA are mainly due to the use of unmonitored field devices and communication protocols with known vulnerabilities [10]. DA systems may be subject to attacks such as Denial of Service (DoS), replay, packet modification, false packet injection, and physical tampering. However, due to the simplicity of field devices in a distribution system, sophisticated security algorithms may be impractical.

As a response to the cybersecurity issues of DAS, some authors have developed vulnerability and risk assessment models using techniques such as attack-defense games [11], and Bayesian attack graphs (BAGs) [12], to quantify the degree of vulnerability present in DA systems. Considering that replay, packet modification and false packet injection are some of the attacks with serious consequences for DAS operation, reference [13] proposes the use of a message authentication code (MAC) and a sync code to guarantee authentication and integrity. A similar concept to provide authentication and integrity is proposed in references [10] and [14]. Nevertheless, these methods are static; they are not designed to detect other forms of attacks such as Denial of Service, or password hacks. Subsequently, such mechanisms may be coupled with intrusion/anomaly detection systems (IDSs/ADSs).

Reference [15] proposes a network-based intrusion detection system (NIDS) which includes deep packet inspection to ensure that benign-looking packets do not create an unstable system state. This adds an extra layer of protection; however, there is an implicit suggestion of the execution of the NIDS on the operators' machine. Thus, the IDS may be centralized, and subject to single point of failures. In reference [16], a technique using μ-PMU measurements is proposed for detecting anomalies. The extensive mathematical operations required suggest that the solution is intended for the operations center.

In spite of the advances in DAS technologies, research on intrusion detection systems, specifically for DAS, is in an early stage. There is also a lack of a response framework for DA cyberattacks. In addition, the current literature is primarily concentrated on DA applications with communication to a central office system. Furthermore, the proposed methods do not defend against coordinated cyberattacks, which may target multiple dispersed nodes. This underscores the critical need for a collaborative distributed approach towards intrusion prevention.

C. Coordinated cyberattack detection and prevention methods

In a coordinated cyberattack, an attacker may use several attack strategies to attack one target, or may attack several parts of one system, or both. The literature on studies pertaining to coordinated cyberattacks is diverse. Reference [17] explores the coordination of physical attacks and cyberattacks on the grid. The authors formulate a bi-level model for coordinating the two types of attacks which minimizes the attack cost according to a predefined budget, while seeking to maximize the reward. Nevertheless, this technique does not provide preventive and mitigation steps when an attack happens.

Reference [18] uses attack templates to formulate correlation indices for attacks. The correlation index is a set of substations that are likely to be attacked based on certain observed attack patterns. The use of optimal power flow to determine the correlation index allows operators to schedule appropriate system reconfiguration and/or load shedding schemes in advance. However, the mechanism is centralized and subject to single point of failures. In addition, the use of extensive math operations restricts its usage in simple intelligent devices.

In [19] the correlation index generator formulated in [18] is combined with an event manager, a correlation knowledge database and a response manager, to detect, correlate and respond to attacks. While the proposed solution is richer in terms of functions, it is subject to the same drawbacks aforementioned concerning the work in [18].

Reference [20] proposes a zero-sum stochastic game approach toward modeling the relationship between the attacker and the defender. It is possible that the attacker may take several steps ahead of the defender while the defender searches for an optimal step. Thus, the defender may take no action, or suboptimal actions, until the end of the attack.

Reference [21] introduces a three-level approach toward mitigating attacks, that includes planning which substations and lines are to be protected, deriving the optimal attack strategy, and the optimal restoration steps as a result of the attack. This paper includes a planning stage, during which cybersecurity resources are optimally allocated.

Reference [22] proposes to use Flexible AC Transmission System (FACTS) devices to periodically perturb the reactance of certain lines in the network. Thus, an attack constructed with outdated reactances can be detected by the bad data detector (BDD).

In [23], the authors propose a method to detect and correlate attacks, using data collected from IDSs installed at different substations. The technique measures correlation according to patterns of abnormal behavior, criticality of substations and the geographical correlation. This mechanism provides correlation of attacks, after they happen. Therefore, it is difficult for operators to predict which substations will be attacked in advance.

The literature on coordinated cyberattacks is heavily focused on the transmission system. Thus, the solutions proposed may be inapplicable at the distribution system level, due to hardware constraints, and the use of methods specific to the transmission system. In addition, the complete delegation of attack response strategies to the human operator is undesirable: in certain scenarios, the human operator may be unable to implement specific defense strategies. An example is when an attacker implements DoS so that the network is unreachable to the operator. Thus, there is a critical need to investigate coordinated attacks at the distribution system level and to formulate solutions in the context of its limitations.

D. Multi-agent systems for intrusion detection and prevention

Traditionally, IDSs can be classified according to their data source or detection technique. According to data source, they may be network-based or host-based. According to detection technique, an IDS may be anomaly-based or signature-based. IDSs have found application in several settings, both in information systems and industrial control systems (ICSs). In [24], the authors propose an anomaly-based IDS that applies a variant of artificial immune system to features of application layer protocols. The work of [25] proposes a multi-model intrusion detection system combining the physical properties of an ICS as well as network properties of its communication infrastructure.

Reference [26] presents a hierarchical intrusion detection system based on multi-agent systems. The agents detect an attack, correlate new attacks with already known ones and determine the extent of similarity. They also plan new attacks, which, when approved by the administrator, are added to the collection of known attacks, resulting in an adaptive approach.

In reference [27], a MAS approach is proposed for detecting attacks and differentiating them from normal faults. A set of three agents operates simultaneously at the same substation, and coordinate with one another using Phasor Measurement Unit (PMU) measurements However, there is a lack of coordination among agents from other substations for detecting and preventing coordinated attacks.

In reference [28], agents are deployed to perform distributed state estimation in their assigned subsystems of the entire network. The requirement to satisfy not only the state equation of the network but also those of the subsystems makes it harder for a False Data Injection (FDI) attack to be concealed.

The solution proposed in [29] is a two-tiered multi-agent hierarchy. Lower level agents are dispersed at different nodes to receive and process data from PMUs. Their data is then sent to a central agent that performs anomaly detection.

II. Motivation

The motivation for this work is to develop an intrusion prevention system (IPS) for DASs. A suitable IPS for DASs should not only be efficient but also:

1) Distributed: The architecture of the proposed systems is preferably distributed to avoid single point of failures in centralized systems.
2) Collaborative: A collaboration mechanism is required of the proposed system in order to correlate attacks and provide protection against coordinated attacks.
3) Light-weight: The hardware used in distribution system nodes tends to have limited computational and storage capabilities. Thus, the proposed solution should be light-weight in order to make it feasible. Although machine learning is useful and can offer accurate results, resource requirements can increase significantly [30]. The intent of this paper is to develop a light-weight intrusion detection system. Machine learning also requires the use of training data. This paper does not assume the availability of such training data.
4) Allow for operator and system intervention: In response to an attack, both the agents and distribution system operators can take mitigation steps to augment each other's effort.

As can be seen from the earlier section, existing work addresses some of the above requirements at a time; a practical solution that addresses all requirements is critically needed. To this end, an approach is proposed in this paper that applies multi-agent systems for cybersecurity of DASs in a novel way to meet the above requirements. The novelty of the paper resides in the fact that:

1) It makes use of a lightweight yet efficient network-based intrusion detection algorithm. The algorithm is based on the DNP3 protocol which is commonly adopted in the U.S.
2) Its use of a multi-agent system is not to detect an attack, as is done in other methods, but to predict the targets of an attack. It therefore leverages the collaboration of agents to protect against coordinated cyberattacks. A consensus protocol, i.e., the link drop max consensus protocol, is formulated to govern the interaction of agents while avoiding the accumulation of unnecessary information.
3) The mechanism of attack correlation is distributed with no hierarchy.

The organization of the rest of the paper is as follows: in section III, the proposed solution is presented. A core element of the proposed solution, the link drop max consensus protocol, is detailed in section IV. The testbed used for simulations is presented in section V, while simulations and results are discussed in section VI. In section VII, the proposed solution is compared with other solutions. Finally, concluding remarks are given in section VIII.

III. MAS for Intrusion Detection

For maximum damage, intruders may attack multiple nodes in the distribution system. First, as an attack model, assume that the motive of the attacker is to disrupt power supply to an area or load of their choice. Assume also, that no social engineering practices are used, so that mainly man-in-the-middle (MitM) attacks are exploited. The set of possible attacks includes replay, denial of service, password hacks, and packet modification/falsification.

An agent is an autonomous software that is able to accept inputs from its environment, process it and take actions based on the outcome. Suppose that a distribution network has an agent installed at each node. The agent, implemented in Volttron [31], is installed on a computing device integrated with remote terminal unit (RTU) functions. Communication-wise, the agents are connected as the nodes are electrically. That is, if node A is connected to node B through a distribution line, then in the communication structure, node A interacts directly with node B. This serves as the initial communication structure of the agents. Also, each agent has three modules: an NIDS module, a prediction module, and a social module.

Again, all agents are assumed to be aware of their own node parameters such as the communication protocol being run (e.g. DNP3, Modbus, etc.), criticality of its load, its neighbors, and software being run by communication devices deployed at the node. The node data is essential for prediction of whether an attack is coordinated, and the targets of the attack.

The proposed Decentralized Intrusion Prevention (DIP), is a four-phase algorithm, depicted in Fig. 1. The phases of the algorithm are explained next.

1) Phase 1: In the first phase, the NIDS module of the agent monitors the local network. The NIDS may be based on any protocol, as the application demands. However, the NIDS implemented in this research is based on DNP3, specifically DNP3 with Secure Authentication v5 (SAv5), as the communication protocol between the operation center and the node. The development of a secure key distribution mechanism for the protocol is assumed to be available. The algorithm for the NIDS is developed to monitor for the following set of attacks from the intruder:

- Flooding (Denial of Service (DoS)): In order to ensure that the remote terminal unit (RTU) at the node is not flooded, the time difference between the arrival of packets is monitored. An alert is triggered when the time interval is lower than a pre-determined threshold value and observation is made F consecutive times.
- Packet falsification/modification: In DNP3 SAv5, critical functionalities such as write, delete, and operate require the receiver to challenge the sender. The sender produces a unique tag through a hashing function, which is sent to the receiver. At the receiver, the same hashing operations are performed and the results compared to the tag received. If they are equal, the identity of the sender is verified and the command is implemented. If the result at the node is not the same as the tag received, the NIDS triggers an alert.
- Replay: In DNP3 SAv5, each critical packet has a challenge sequence number. The sequence number is checked to ensure that already received numbers are not repeated. In the proposed NIDS, an alert is raised when the challenge sequence number of a received packet is smaller than or equal to the latest recorded at the node.
- Brute-force password hack: By successfully logging in, the adversary may be able install/delete applications, and/or run commands. The login credentials are assumed unavailable to the attacker. Hence, in the proposed NIDS, P consecutive failed password attempts to log in is flagged.

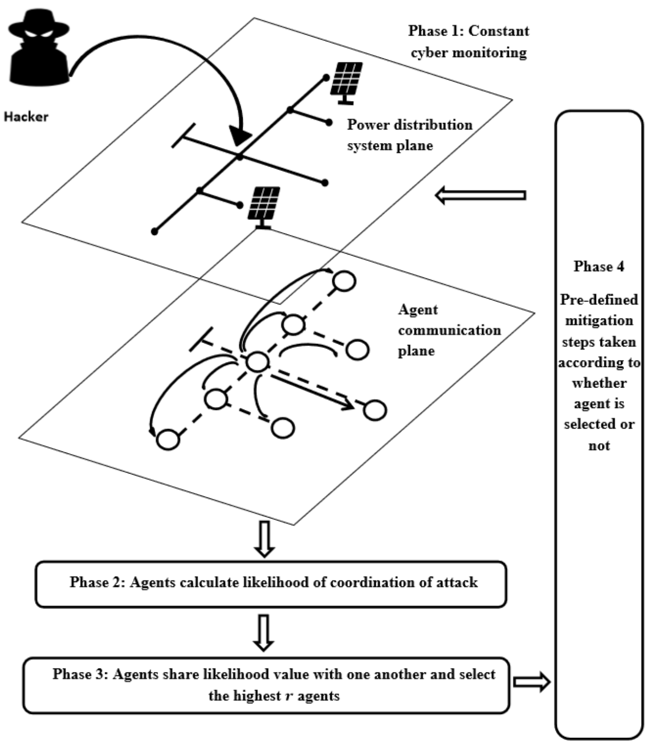
Phase 1: Constant cyber monitoring
Hacker
Power distribution system plane
Agent communication plane
Phase 4
Pre-defined mitigation steps taken according to whether agent is selected or not
Phase 2: Agents calculate likelihood of coordination of attack
Phase 3: Agents share likelihood value with one another and select the highest r agents

Fig. 1: An illustration of the proposed DIP

The NIDS is able to accurately detect configured attacks. For instance, consider a complicated attack where the adversary copies the unique tag of a packet in transit and attaches this to a fabricated packet. The NIDS at the node is able to detect this. Indeed, at the node, the hashing operation is performed over the entire received packet, and the results compared to the unique tag. This attack succeeds only when the attacker is aware of the correct hash key. In this paper, sharing of all cryptographic keys has been assumed secure and confidential.

All agents continually monitor their own cyberspace in the first phase, using the proposed NIDS. A transition is made into the next phase if and when an intrusion is detected. Suppose an intrusion is detected at one of the nodes in phase 1. The node is required to broadcast an alert to the operations center and to all agents in the network. The alert contains the following: time stamp of the detected intrusion, ID of the reporting agent, suspected attack type, other descriptive data such as the target device, and the protocol being run.

The broadcast alert is encapsulated in a TCP/IP packet. A hash digest is also added to the message to provide authentication. It is assumed that the key for this hashing operation is updated when session keys for the communication between nodes and the operation center are updated. Therefore, for securely exchanged and updated keys, the attacker is unable to falsify or modify an alert. Having received the alert, each recipient agent enters the second phase.

2) Phase 2: Each recipient agent needs to predict with some certainty whether an attacker will target its node. This is done by measuring its correlation with an alerting node (agent). The proposed correlation model uses three factors: attack patterns, criticality of load at the node, and software correlation. An attacker may be attempting different attack techniques, such as replaying an already captured packet at one node, while modifying the same at another node. Thus, the observation of attack patterns is a good indicator of correlation. In addition, nodes that serve critical loads are by nature attractive to attackers. Subsequently, in a coordinated attack, criticality of load at a node is an indicator of correlation. Lastly, an attacker may leverage vulnerabilities that may be present in firmware and other software run by the intelligent device. In this case, targeted nodes may not be correlated according to load type or attack patterns, but according to the software they run. In this paper, these three factors are used to measure correlation, and consequently used to predict the likelihood that an attacker will target a recipient node. The measure of correlation is obtained by calculating three indices to quantify each of the three factors: intrusion potential, criticality index, and software and operational vulnerability index.

(i) Intrusion potential (ν): This is the likelihood that some observed pattern evolves into an intrusion. The received report is compared to current security logs from the past h units of time. h is a user-defined variable. Let there be a set of attacks $\mathcal{S} = \{d, f, r, p\}$ where d, f, r, p represent flooding (DoS), packet falsification, replay, and attempted password login, respectively. For an attack $x \in \mathcal{S}$, let x_t be a threshold value and x_r be its maximum recorded normal occurrence in the security logs at the recipient node. Also, let ω_x be the weight assigned to that attack. The weight is chosen according to the following rules:
   a) if the attack was not reported and its occurrence in security logs at the recipient station is below half of its threshold, $\omega_x = 1$,
   b) if the attack was not reported but its occurrence in security logs at the recipient station is greater than or equal to half of its threshold, $\omega_x = 2$,
   c) for a reported attack, if its occurrence in security logs at the recipient node is below half of its threshold, $\omega_x = 2$, and
   d) for a reported attack, if its occurrence in security logs is greater than or equal to half of its threshold, $\omega_x = 3$.

TABLE I: Criticality indices for different nodes.

Node type	Criticality index (κ)
Node supplying highly critical load (e.g. hospitals, critical infrastructures such as water supply)	0.9
Node supplying critical load (e.g. Important industry and commercial load)	0.6
Node supplying non-critical load (e.g. homes)	0.3

The intrusion potential is then given by:

$$\nu = \frac{\sum_{x \in \mathcal{S}} \omega_x (\frac{x_r}{x_t})}{\sum_{x \in \mathcal{S}} \omega_x} \tag{1}$$

(1) is a weighted average of ratios which correlates log patterns to a reported attack. In order to capture the possible use of different attack techniques at a time, (1) considers log patterns for all types of attacks in the attack set. For example, an attacker may send a command to open the load switch at a node through replay or a falsified packet, and immediately follow up with a flooding attack in order to prevent the node from sending status information to the operations center.

(ii) Criticality index (κ): In the algorithm, criticality indices are set according to Table I.

(iii) Software and operational vulnerability index (ψ): The software vulnerability index measures the extent to which the affected device of the reporting node is related to that of the recipient node. The inclusion of this index is to capture similar software and/or operational vulnerabilities that an attacker could be leveraging. Set $\psi = 1$ if both devices are of the same manufacturer and use the same software versions, $\psi = 0.66$ if they are of the same manufacturer but use different software versions, $\psi = 0.33$ if they are of the same manufacturer but different software, and $\psi = 0$ if they are not related.

Note that a node may be correlated to an alerting node in one or more of the three factors. An accurate decision is made when all factors are considered. A weighted sum of the indices takes into account all three factors and captures the overall correlation between the recipient node and an alerting node. Subsequently, having found the three indices, an empirical judgement is made as follows:

$$\rho = w_\kappa \kappa + w_\nu \nu + w_\psi \psi \tag{2}$$

$$w_\kappa + w_\nu + w_\psi = 1 \tag{3}$$

$$w_\kappa, w_\nu, w_\psi \geq 0 \tag{4}$$

The symbols w_κ, w_ν and w_ψ are non-negative weights assigned to the different indices. They are normalized by equation (3). The value ρ is a prediction made by the recipient agent regarding the extent to which it believes it will be a target for an attacker. Clearly, ρ is affected by the values of κ, ψ and ν which are assigned based on how the features of the receiving node compare with those of the alerting node. It is also affected by the weights w_κ, w_ν and w_ψ, and it is necessary to choose the weights in order to maximize the accuracy of this prediction. An experiment on tuning the weights is provided in the simulations presented in this paper.

3) Phase 3: In this phase, agents share their judgment values, ρ, with their neighbors, encapsulating them in TCP/IP packets. This phase is implemented by the social module of the agent. A neighbor of a node is any node which has a direct communication link to it. Having received their neighbors' values, each agent selects the highest r of them, together with the IDs of the agents whose values are selected. The selected list is shared again, and the process repeated. At each sharing stage, the current selected list is compared to the previously sent list. If the two are equal, an agent does not re-share as this will be repetitive. By doing so, network resources are reserved for nodes that have new data to share. The process is repeated until each agent has the same list of agent IDs and their corresponding values, which are indeed the highest in the network.

The proposed consensus protocol, the link drop max consensus protocol, belongs to the family of max consensus algorithms [32]–[34], in which the maximum initial value in the network is sought. However, the proposed protocol differs in the check for repeated information.

4) Phase 4: In this phase, agents take specific mitigation actions based on the outcome of phase 3. If an agent's value is selected, it enters a protective mode for a time period T. The agent that broadcasts the alert also enters protective mode. In protective mode, all remote control functionalities are disabled. This allows the operation center time to attend to the security issues that arise. On the other hand, if an agent is not selected, all NIDS thresholds (d_t, p_t, etc.) are set to $n\%$ of their initial values. Should another alert be broadcast within t units of time from a previously received one, all agents enter the protective mode. T, t, and n are user-defined variables.

It should be noted that when a node undergoes an attack, say a DoS attack, and broadcasts this, it enters protective mode after the first round of the algorithm. Thus, before T elapses, the node cannot be attacked again since it is isolated from further communication. A node in protective mode continues to function electrically and may send periodic status information to the operations center; only outgoing traffic from the node is allowed. This is different from an explicit ingress filtering scheme implemented to mitigate attacks, as discussed in [25]. In such schemes, packets from only the offending address are dropped. While this method may serve to selectively block an attacker's traffic so that the operator is still able to communicate with the node, a skillful attacker who sends packets with varying spoofed addresses may escape the check.

IV. The link drop max consensus protocol

The link drop max consensus algorithm allows a group of distributed agents, each with some value, to share and select the maximum value amongst them without repeating already shared information. In this section, the terms "algorithm" and "protocol" refers to the link drop max consensus protocol. It is assumed that there is no data loss in the network.

Premise: Nodes in the distribution grid are not always communicating but only initiate a conversation when a decision is to be made. They are made aware of a decision to be made when one of the nodes informs them to that effect. The communication topology starts with an initial structure that changes as the conversation progresses. The initial structure is identical to the topology of the distribution system, and both are assumed radial. It is also assumed that there are no islands. In the initial network, every pair of connected nodes has a two-way link between them.

From these, the initial model of the distribution network is a strongly connected digraph $\mathcal{G}$ with a set of vertices $\mathcal{V}$ and a set of extraverted edges $\mathcal{E}$. The vertices represent the nodes while edges represent the active communication links between the nodes. Hence, 'vertex', 'agent' and 'node' are used interchangeably, as are 'edge' and 'link'.

Thus, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a non-empty set consisting of m nodes. There is no loop in the graph, i.e., there is no single edge that connects a vertex to itself.

The graph possesses an adjacency matrix $\mathbf{A}$, which is a matrix with elements $a_{ij} = \{0, 1\}$. An element a_{ij} is 1 if node i communicates to node j, and 0 if it does not. It should be noted that $a_{ij} = 1$ necessarily means $a_{ji} = 1$ for the initial network, but is not guaranteed for all time. The degree matrix of the graph is the diagonal matrix $\mathbf{\Delta}$ whose diagonal elements are the total number of edges attached to a node. The neighborhood of a node $i \in \mathcal{V}$ is given as $\mathcal{N}_i := \{j : a_{ij} = 1\}$.

By nature, the state of the graph for max consensus protocols does not evolve according to a set of linear dynamic equations as is the case for the average consensus protocol (ACP) [35]. Rather, the proposed algorithm follows a sequence of discrete logical steps. Assume that nodes in the grid are selecting the max value among all the judgement values calculated (in phase 2) by the nodes in the grid.

1) At the start of a conversation, each agent i in the distribution network has an initial value $\rho_i(0)$ and an initial ID $z_i(0)$. The initial ID is the ID configured for the node. To standardize notation, let x represent ρ going forward. The initial state vector of the graph is $\mathbf{x}(0)$, and $\mathbf{A}(0) = \mathbf{A}^\top(0) \in \mathbb{R}^{m \times m}$.
2) Each agent i also possesses a vector $\mathbf{p}^{[i]}(k) \in \mathbb{R}^{|\mathcal{N}_i|}$ in memory which stores the values of its neighbors, and a vector $\mathbf{u}^{[i]}(k) \in \mathbb{R}^{|\mathcal{N}_i|}$ which stores their IDs. At the start of the conversation, $\mathbf{p}_j^{[i]}(0) = -\infty$ and $\mathbf{u}^{[i]}(0) = z_j(0)$ for $j \in \mathcal{N}_i$.
3) The information shared by agent i in event k is a concatenation of the agent's selected value and its ID, denoted by $\sigma_i(k) = \{x_i(k)|z_i(k)\}$.
4) Agent i shares $\sigma_i(k)$ with its neighbors if $a_{ij}(k) = 1$, for $j \in \mathcal{N}_i$.
5) For $i \in \mathcal{V}$, $x_i(k+1) = \max\{\|\mathbf{p}^{[i]}(k)\|_\infty, x_i(k)\}$. That is, the next state of node i is the maximum of all received values, including its own. The agent ID of the selected value is also stored as $z_i(k+1)$. Thus, $\sigma_i(k+1) = \{x_i(k+1)|z_i(k+1)\}$.
6) Following the above steps, the adjacency matrix is updated as follows. For all $i \in \mathcal{V}$, $j \in \mathcal{N}_i$,

$$a_{ij}(k+1) = \begin{cases} 1 & \text{for } x_i(k+1) - x_i(k) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Step 6 is essential in order to avoid repetition and subsequent information accumulation. Since the graph starts with a two-way link between every connected pair of nodes, the dropping of links by a node only makes those links one-way. The node, therefore, may still receive information from its neighbors but does not communicate back unless the update rule indicates so. Consequently, an agent may re-establish already dropped links.

The proof of convergence of the algorithm, as well as its speed, is presented in the appendix. It is shown that the speed of convergence is bounded above by the diameter D of the network graph.

A. Advantage of the link drop max consensus protocol

Collisions are stochastic events that may occur in a given communication network. However, repeated sharing of redundant information between neighbors unnecessarily exposes the link to collisions. This is especially true for heavily constrained networks. When a collision occurs on the link between neighbor agents, a retransmission is required, which potentially increases the time required to achieve consensus.

The likelihood of collision occurring is dependent on the probability that a link is used at the same time by neighbors. Therefore, it is reduced when the probability of a neighbor communicating decreases. This is the unique feature of the link drop max consensus protocol, when compared to other max consensus protocols, e.g., [32]–[34]; it avoids the sharing of redundant information.

In the link drop max consensus protocol, the probability that an agent has the max value, and subsequently will not communicate in the next state is $\frac{1}{n_c(k)}$ where $n_c(k)$ is the number of communicating nodes in the kth state. Considering that there is convergence, it follows that $n_c(k) \to 1$ as $k \to D$. Let the probability that a neighbor communicates in the kth state be $p_c(k)$. In the original max consensus protocol, $p_c = 1$ for all time during agent communication. However, in the link drop max consensus protocol, $p_c = 1$ only in the first state. In subsequent, states $p_c(k) = 1 - \frac{1}{n_c(k)}$. Hence, $p_c \to 0$ as $k \to D$.

For simplicity of illustration, assume a circular graph with $V = 13$ nodes, implementing the link drop max consensus protocol. All agents communicate in the first state ($k = 0$); $p_c = 1$. For the second and third states, $p_c = 1 - \frac{1}{V}$ and $p_c = 1 - \frac{1}{V-1}$ respectively. For subsequent states $k = 3, ..., D$ in the circular graph, $n_c(k) = V - 1 - \sum_{k=3}^{D} 2(k-1)$. Therefore, $p_c(k)$ is given by:

$$p_c = 1 - \frac{1}{V - 1 - \sum_{k=3}^{D} 2(k-1)} \tag{5}$$

Figure 2 shows a plot of p_c against state k for a circular graph with 13 nodes.

From this, it is shown that the link drop max consensus protocol reduces the probability of agent communication over time, and therefore reduces the likelihood of collisions over time.

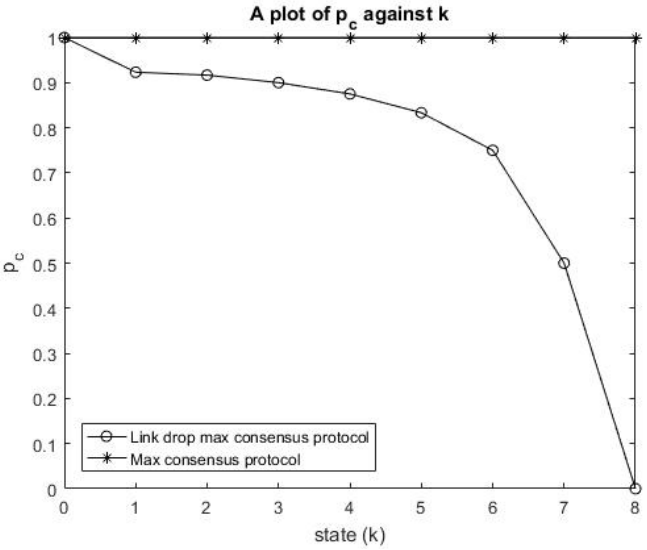

Fig. 2: Comparing variation in p_c with state in the link drop max consensus protocol and the generic max consensus protocol for a 13-node circular graph

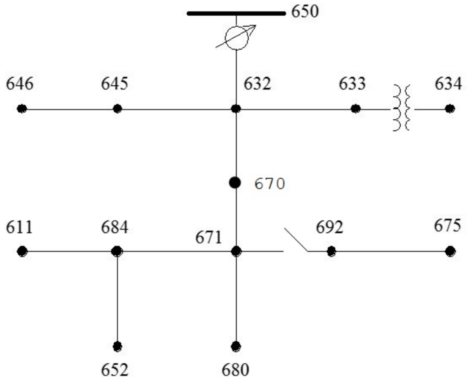

Fig. 3: IEEE 13-Node test feeder

V. Setup for validation

The IEEE 13-Node Feeder (shown in Fig. 3) is used for testing and validation. In the setup, nodes 671, 675, 680 and 692 serve highly critical load; nodes 632, 633, 645 and 646 serve critical load; and nodes 611, 634, 652, 670 and 684 serve non-critical load.

Agents are developed using Volttron, a Unix-based Pacific Northwest National Lab (PNNL) open-source agent-based platform [31]. Agents are developed and run on a virtual Linux-Mint system. These agents form part of the cyber model of the grid and are initially connected to communicate with one another as the nodes are connected electrically.

Scapy, an open source packet manipulation tool [36], is used. Originally, Scapy has no DNP3 library. A DNP3 library with SAv5 functionality is built to extend its capabilities for this test. An attack simulation platform is also developed using the added DNP3 library of Scapy to implement the different attacks enlisted in this paper, completing the cyber model.

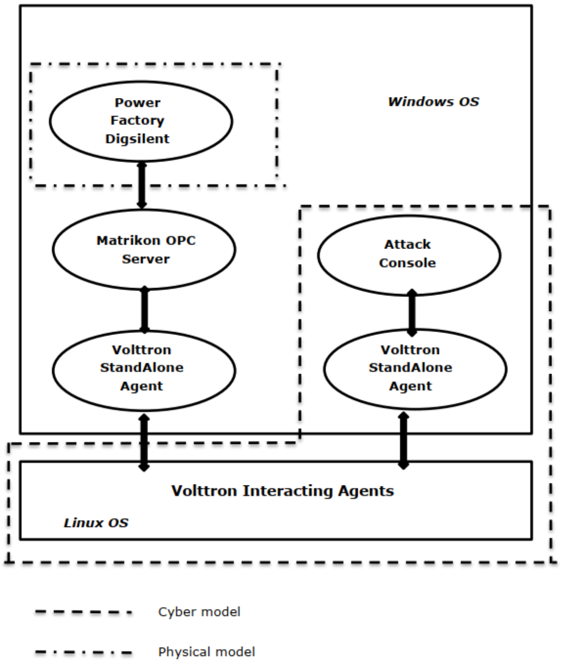

Fig. 4: A Cyber-power system simulation setup

The power system model is built using Power Factory DIgSILENT, an industry level power system simulation tool. DIgSILENT is run on Windows 10 operating system. Matrikon OPC server is used to connect the cyber and power system models in a real time environment. The interaction among these components is shown in Fig. 4.

VI. Simulation and results

Two studies are performed in the simulation. In the first study, some properties of DIP are investigated, while in the second, the performance of DIP under coordinated attacks is assessed.

A. Study 1: Investigating some properties of DIP

The efficiency of DIP in detecting attacks is dependent on that of the NIDS implemented in the first phase. Nevertheless, the accuracy of predicting the correlation of attacks is dependent on the relational weights used in finding the judgement value in the second phase. In surveyed papers on coordinated cyberattacks, authors do not explicitly assess the accuracy of suggested techniques for determining correlation. In this study, such an investigation is conducted.

1) Efficiency of NIDS implemented in Phase 1: The performance of the NIDS in phase 1 is measured by the false positive and false negative ratios. A smaller ratio is desirable. The false positive ratio (FPR) is the proportion of normal packets that are misclassified. For the implemented NIDS, the FPR may be determined from the probability that an authorized user enters the wrong credentials P times, the probability that the operation center sends replayed packets or packets with incorrect hashes, and the probability that the operational center sends a cluster of packets greater than what is used in detecting flooding.

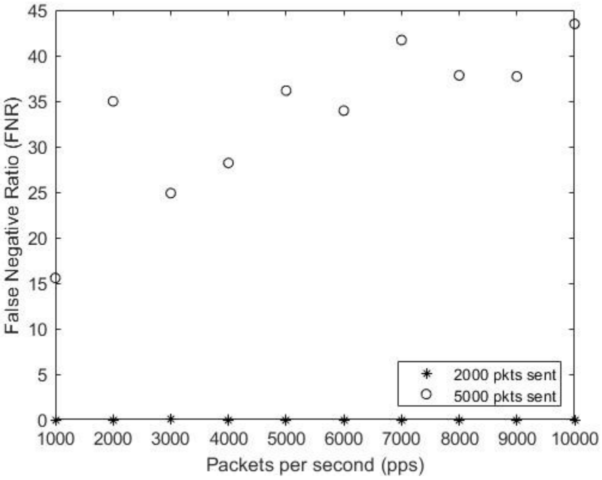

Fig. 5: FNR with varying packet rates for different total packets sent

The false negative ratio (FNR) is the proportion of abnormal packets that are misclassified. For the proposed NIDS of phase 1, the FNR may be determined by comparing how many alerts are generated with how many malicious packets are sent. To measure the FNR, the NIDS was installed on the slowest computer available, which is an Intel Core i3 computer. The plot in Fig. 5 shows the FNR for when 2000 packets and 5000 packets are sent at varying rates.

The highest FNR, 45%, occurs when 5,000 packets are sent at a rate of 10,000 packets per second (pps). The FNR appears to increase with increasing number of packets sent. It is also observed that when 5,000 packets are sent at 7,000 pps, the FNR is close to 45% and higher than that obtained at 8,000 pps and 9,000 pps respectively. This indicates that flooding at a higher rate is not necessarily guaranteed to result in higher FNR. When implemented on a real RTU, the FNR may be higher.

2) Effects of relational weights on the success rate of DIP: In phase 2, DIP predicts the motive and/or leverage of the attacker, and therefore predicts their target nodes. The accuracy of this prediction is termed the prediction success rate (PSR) of DIP. The PSR is the proportion of targeted nodes that enter protective mode after the first detection of intrusion. For instance, if the attacker is aiming for critical nodes, and two, instead of all four, entered protective mode at the end of phase 4, then the PSR is 50%. The PSR is dependent on the judgement value ρ. As long as a node has one of the highest three ρ values, it is guaranteed to enter protective mode. It is therefore desired that at the end of phase 2, nodes that will be targeted by the attacker have the highest judgement values.

The judgement value ρ is in turn determined by the probabilistic indices κ, ν, ψ and the preset weights w_κ, w_ψ, w_ν. As indicated in an earlier section, the values assigned to the indices depend on the correlation between the features of the alerting node and those of the recipient node. The PSR is therefore dependent on: (i) the correlation of features among nodes, and (ii) the preset weights. Consider two networks A and B, adapted from the IEEE 13-Node Test Feeder. Let all the

TABLE II: Distribution of features in network B

Features	Nodes
Nodes supplying highly critical load	680, 692, 671, 675
Nodes implementing fictitious software ABC OS9 1.0.1	611, 634, 671, 652
Random nodes to be attacked	632, 645, 652, 692

critical nodes of network A run the same fictitious software, while those of network B run different software. Assuming an attacker targets the critical nodes, then for the same set of weights and the same attack potential, the critical nodes of A are more likely to enter protective mode than those of network B. This is because even though all critical nodes will have the same $\kappa = 0.9$, those of network A will also have $\psi = 1$, while those of network B will have different values of $\psi < 1$. Thus, the PSR of network A is higher than that of network B, for the same weights. In order to increase the PSR of network B, the weights must be changed. Therefore, for every network, the preset weights need to be well chosen in order to achieve its maximum PSR.

Algorithm 1 details a tuning procedure for choosing the weights. Let nodes 671, 675, 680, and 692 of the IEEE 13-Node Test Feeder be highly critical nodes, and let them all run a fictitious software ABC OS9 1.0.1. Using Algorithm 1, a weight w is varied from 0.1 to 0.9 while keeping the other weights at $0.5(1 - w)$ each. For each variation of the weight, $\mathcal{M} = 100$ attacks are performed. The resulting PSR is recorded and a curve is plotted, as indicated in step 10 of the algorithm. This is repeated for each of the weights. Fig. 6 shows the curves obtained, and the common PSR that preserves equation (3) is approximately 85%. At the end of the tuning algorithm the weights are chosen as $\{w_\kappa, w_\psi, w_\nu\} = \{0.1, 0.282, 0.618\}$ and the resulting PSR is 86%.

Assume a second network adapted from the IEEE 13-Node Test feeder. The features of the nodes are set according to Table II. Using the same algorithm, the weights are determined as $\{w_\kappa, w_\psi, w_\nu\} = \{0.406977, 0.2825, 0.310523\}$. The PSR recorded is 61.33%. To illustrate the strengths of DIP, even with a network of low PSR, the node features in Table II and the weights found for this configuration are used for the second study.

B. Study 2: assessing the performance of DIP

In this section, a base case in which there is DAS without inter-node communication, as is the case in DIP, is first implemented. Next, DIP is simulated under a sequential coordinated attack, and under a concurrent coordinated attack. Altogether, three scenarios are simulated. In all simulations, the following thresholds are used: $d_t = 200$, $f_t = 1$, $r_t = 1$, and $p_t = 5$. The replay and packet modification thresholds are chosen with the assumption that the operation center does not send badly crafted packets. The password threshold is chosen according to standard industry practice. The flooding threshold is chosen such that it is above the average cluster sent by the operations center. Also a flooding alert is triggered much earlier before malicious packets are missed, according to FNR patterns.

Algorithm 1 A tuning algorithm for determining the relational weights

```
 1: Let $W = \{w_\kappa, w_\nu, w_\psi\}$
    LOOP Process
 2: for each $w \in W$ do
 3:     $w = 0.1$
 4:     while $w \leq 0.9$ do
 5:         Set $w_n = 0.5(1 - w)$ for all $w_n \in W \setminus w$
 6:         Perform $\mathcal{M}$ attacks corresponding to $w$
 7:         Record the average PSR
 8:         $w = w + 0.1$
 9:     end while
10:     Plot a curve of average success rate against $w$
11: end for
12: Determine the common PSR that preserves equation (3)
    LOOP Process
13: for each $\{w_1, w_2\} \in W$ do
14:     Determine the intersection of their curves with the
        minimum success rate line
15:     Set $w_3 = 1 - w_2 - w_1$ for $w_3 \in W \setminus \{w_1, w_2\}$
16:     Perform $\mathcal{N}$ attacks with weights $\{w_1, w_2, w_3\}$ and
        record the average success rate
17: end for
18: Select $\{w_\kappa, w_\nu, w_\psi\}^*$ which gives the highest PSR
```

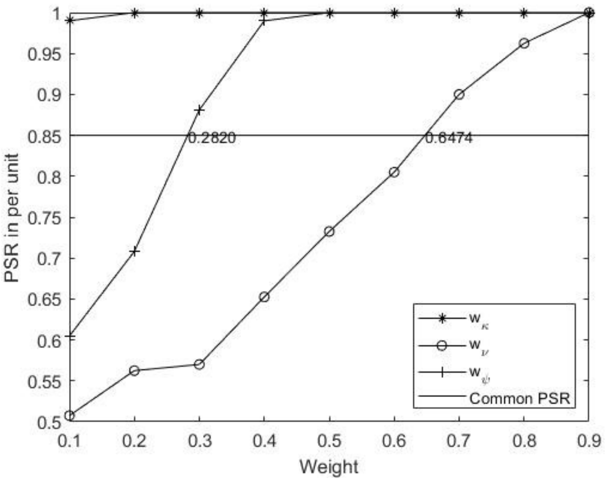

Fig. 6: Plots of PSR against varying weights, as obtained from Algorithm 1

Here in this study, the entire network, with all agents, is run on an Intel core i7 computer. Also, agents are set to select the agents with the maximum 3 values in addition to the alerting node. Since the simulation is run on a single computer, the communication between agents is considered synchronous. The diameter D of the test feeder is 6. Thus, using a time unit of 0.5s according to observed processing speed of the computer, it is determined that the upper bound to convergence is 3s. In the event that an alert is detected, all agents not in protective mode reduce their thresholds to $n = 50\%$, rounded up to the nearest whole number.

The adversary has already performed reconnaissance on the

TABLE III: Sequence of events in scenario 2

Event	Attack	Time Stamp	Result	Time Stamp
1	Replay attack launched on node 692	7:06:27,521	Attack detected	7:06:27,668
2	Agent at node 692 broadcasts alert	7:06:27,764	Agents calculate ρ and enter into conversation	-
3	Agent conversation ends	7:06:30,993	Agents 646, 671, 680, 692 in protective mode. All others have reduced thresholds	7:06:30,993
4	Attacker attempts to log in to node 692	7:07:24,693	No log in console available, node in protective mode	-
5	Attacker attempts to log in to node 675	7:08:35,874	After three unsuccessful attempts, attacker is blocked	7:09:06,314
6	Alert is broadcast	7:09:06,400	All agents in protective mode	7:09:06,771

grid and has a good knowledge of which nodes to attack. Their motive is to disrupt power supply to the highly critical load.

1) Scenario 1: DAS without inter-node communication: In the base case, a NIDS with functions described in phase 1, is installed at each node. However, there is no inter-node communication. The IDS, nevertheless, alerts the control center of suspicious activities. At 06:57:43,341, the adversary begins attacks at node 692 since this node serves a critical load. Five password login attempts are made. The NIDS at the node flags this and blocks the attacker. The attacker then attempts to log in to nodes 671, 675, 680, 645 and 646 in succession. Finally, they are able to log in to node 632 on the third count (recorded at 07:03:34,231), and run a command to disconnect the load. Even though there are NIDSs at each of the nodes in the network, the attacker is still able to launch an attack due to the large attack surface. Thus, it is observed that the installation of non-interacting NIDSs does not provide maximum cybersecurity.

2) Scenario 2: DAS with node communication using DIP under sequential coordinated cyberattack: In the second scenario, DIP is implemented. Table III shows the sequence of events and results in this scenario. It is observed, in the first event, that after the first attack is detected at 7:06:27,668, the attacked node (node 692) broadcasts an alert. All agents come to a consensus approximately 3s after the alert is received at the nodes. Nodes 692, 671, 680 and 646 enter protective mode. Node 692 is unresponsive to further attacks conducted in event 4. Having found three highly critical nodes in protective mode, the attacker further attempts to log in to node 675. However, after 3 unsuccessful attempts (due to a reduction of thresholds at the end of event 3), an alert is broadcast. At 7:09:06,771, all agents enter protective mode, an average of 371ms after the second alert is broadcast. Compared to the first scenario, DIP significantly reduces the effective attack surface of the distribution system, and the performance is validated.

3) Scenario 3: DAS with node communication using DIP under concurrent coordinated attack: In this scenario, the system implemented in scenario 2 is used. The adversary attacks two nodes at the same time. The attacker has already captured a packet meant to open a switch at node 675 and intends to replay this. Meanwhile, there is also an attempt to log in to node 671 using a password hack. At 15:26:43,176, node 671 detects the consistent password attempts and flags this. An alert is sent to all other nodes and to the control center. Agents start calculating their judgement values. However, at 15:26:43,264, less than 100ms later, node 675 receives a replayed packet and flags this immediately. It broadcasts an alert to all agents and to the control center. Nodes begin to abort inter-node communication at 15:26:43,331. At 15:26:43,695 all nodes are in protective mode.

VII. Comparison with other work

In this section, the proposed DIP for DASs is compared with related work in the literature.

A. Comparison with Deep Packet Inspection

First, the NIDS implemented in DIP (henceforth referred to as NIDS-DIP) is compared with an NIDS similar to that provided in [15] (a REFerence method henceforth referred to as NIDS-REF). NIDS-REF, among others, performs deep packet inspection to ensure that packets adhere to accepted operational procedures. To achieve an objective comparison, NIDS-REF is developed according to the DNP3 SAv5 protocol, even though the original implementation in [15] is based on Modbus. The following attacks are executed: flooding, packet modification, replay, password hack, invalid command, and breach of operational procedure.

The detection results show that NIDS-DIP is able to detect and correctly alert on flooding, packet modification, replay, and password hack. This is expected as it has been configured to monitor for these attacks. However, for invalid command and breach of operational procedure attacks, NIDS-DIP detects the presence of attacks but alerts these under a different category. This is due to the use of integrity and authentication checks in DNP3 SAv5. An attacker's packet that contains invalid commands or breaches operational procedure also fails integrity and/or authentication check.

NIDS-REF accurately detects and alerts on flooding, invalid command, and breach of operational procedure attacks. Password hacks are undetected since it is not configured to monitor for such attacks. NIDS-REF is not configured to monitor for packet modification and replay, hence, it does not alert on these. However, due to the use of DNP3 SAv5, a packet that fails authentication and/or integrity checks is dropped and an error message logged.

NIDS-DIP performs intrusion detection using network features of a received packet; it does not perform deep packet inspection. While this makes it easier to implement and lighter to install, it is unable to detect insider attacks as well as attacks from hackers with insider details.

TABLE IV: Comparing DIP with CENTRAL-REF

	Offline Computation Time (hrs)	Online Attack Correlation Time (s)	Prediction Success Rate (PSR) (%)
DIP	4	3.325	61.33
CENTRAL-REF	0	0.48	41.33

B. Comparison with Centralized Correlation

Secondly, the decentralized correlation technique of DIP is compared to the centralized correlation method proposed in [23] (a REFerence method henceforth referred to as CENTRAL-REF). CENTRAL-REF originally correlates a location index, critical index and an abnormal behavior index in an iterative matrix multiplication technique. Once a steady state correlation vector has been obtained, the maximum value is selected as the correlation index. Moreover, the technique correlates already detected attacks in several substations to establish whether they are correlated or not. However, to achieve an objective comparison, the method has been adapted to predict the targets of an attack. This is done by applying the technique to correlate the three indices found for each node.

The tests are performed on the network with node features shown in Table II. This is the same network used in studies in subsection VI-A2 of section VI. Table IV summarizes the outcome.

The offline computation time required in DIP is to determine the values of the relational weights w_κ, w_ν and w_ν. In CENTRAL-REF, no such precomputation is required. The online attack correlation time of DIP is also expectedly longer than that of CENTRAL-REF. This is primarily due to the time complexity of the consensus algorithm employed in DIP, which is absent in centralized systems such as CENTRAL-REF. The PSR of 61.33% for DIP is same as that which is obtained under subsection VI-A2 of section VI. Nonetheless, the PSR of DIP is found to be higher than that of CENTRAL-REF. This is due to the fact that CENTRAL-REF applies equal weights to the indices. Thus, for the same network features, CENTRAL-REF tends to have a lower PSR than DIP.

VIII. Conclusion and Future Work

In the first study, when 5,000 packets are sent with an interval of 0.1ms between packets (i.e. 10,000 pps), about 45% of the malicious packets are missed, compared to 0% for 2,000 packets. The FNR is observed to increase as the total number of sent packets increases. While this is expected, it could be improved. Even though the FNR is dependent on the specifications of the machine on which it is run, the use of data storage- and processing-efficient mechanisms impact the performance. Future work, therefore, ought to investigate the use of such mechanisms as the Bloom Filter to improve the processing and storage efficiency of the NIDS. In addition, the NIDS ought to be updated to include other signatures that evolve with time.

Furthermore, the relational weights of phase 2 directly impact the success rate of DIP. These weights may also vary from one network to another since the nodes in different networks may have different features. There is currently no universal rule to finding these weights and they must be tuned experimentally using initial offline simulations of the network. While high PSRs may be obtained for certain networks, it is believed that making the weights adaptive (such that each node chooses its own set of weights), and inclusive of the cyber history of the network, would help to improve the PSR for any given network. Subsequently, future work will investigate this.

From the second study, one of the main benefits of the algorithm illustrated is that only two nodes are available from the attacker's perspective. This drastically reduces the effective attack surface. For an N-node system, assuming all nodes are equally vulnerable, this is a reduction by $1 - \frac{2}{N}$. For the 13-node test feeder, this is approximately 84.7%.

Again, the attack reward is substantially reduced. This is because the algorithm takes into account coordinated attacks and subsequently attempts to capture the motive or leverage of the attacker. As can be seen from Table III, by detecting an attack at node 692, three of the four critical nodes enter a protective mode and are hence out of the reach of the adversary.

This solution is scalable; operators only need to update the node data of an agent (such as the list of neighbors) when changes are made. Nevertheless, when applied to a network larger than the test system used in this paper, the number of agents will increase, as an agent is required at each node of the network. This implies an increase in cost of installation and maintenance. Thus, an investigation into the optimal number and location of agents is an important task for the future work. In addition, a larger network may possess a larger diameter, implying that agents would take longer to reach a consensus (Phase 3). Consequently, the use of graph partitioning mechanisms that reduce the effective diameter of the network should be investigated.

Appendix A
Convergence of the link drop max consensus algorithm

The conditions for convergence are:

1) A consensus is reached: for any initial state of the graph $\mathbf{x}(0)$ there exists a value $x_b \in \{\mathbf{x}(0)\}$ where $x_b = \max\{\mathbf{x}(0)\}$, possessed by agent with initial ID z_b such that $\lim_{k\to\infty} x_i(k) = x_b$ and $\lim_{k\to\infty} z_i(k) = z_b$ for all $i \in \mathcal{V}$.
2) Nodes are no longer communicating unless another decision process is started. That is, $\mathcal{E} = \emptyset$.

In the sub-sections that follow, convergence of the algorithm has been shown, first for the case where the maximum value is being chosen and for when r highest values are being chosen (such as ranking of the 3 highest judgement values).

A. The case for one value

Assume that there is a distribution grid in which each node i has calculated some unique initial value $x_i(0)$. This is the first graph state $\mathbf{x}(0)$. Now let the maximum value be x_b and the node that possesses x_b be node b with initial ID z_b. At the beginning of each cycle, a node may drop the link between itself and its neighbors. The probability for this is dependent on the outcome of the update rule of the adjacency matrix from the previous cycle. Let the probability that the ith node will drop links in the kth cycle be $p_i(k)$. Then, at the end of the first cycle for node b, $p_b(2) = 1$ since $x_b > x_i$ for all $i = 1, ..., b-1, b+1, ..., m$. Since $x_b(1) - x_b(0) = 0$, $a_{bj}(2) = 0$ for all $j \in \mathcal{N}_b$ in the second cycle.

In the first cycle, all neighbors of node b received x_b. Thus, in the second cycle, $x_j(2) = x_b$, $p_j(2) = 0$ and $a_{jt}(2) = 1$ for all $j \in \mathcal{N}_b$ and all $t \in \mathcal{N}_j$. The neighbors of node b re-share x_b with their own neighbors. In the third cycle, $x_j(3) = x_b$, $p_j(3) = 1$ and $a_{jt}(3) = 0$ for all $j \in \mathcal{N}_b$ and all $t \in \mathcal{N}_j$.

Thus as $k \to \infty$,

$$x_i(k) = x_b \quad \forall \quad i = 1, ..., m \tag{6}$$

$$p_i(k) = 1 \quad \forall \quad i = 1, ..., m \tag{7}$$

$$a_{ij}(k) = 0 \quad \forall \quad i, j = 1, ..., m, i \neq j \tag{8}$$

$$\mathbf{\Delta}(k) = \mathbf{0} \tag{9}$$

$$\mathbf{A}(k) = \mathbf{0} \tag{10}$$

The Laplacian matrix is therefore:

$$\mathbf{L}(k) = \mathbf{\Delta}(k) - \mathbf{A}(k) = \mathbf{0} \tag{11}$$

The Laplacian is now a zero matrix, and has 0 eigenvalues with multiplicity m. Also, $\mathcal{E} = \emptyset$. There is no more active communication between any two nodes. Thus, the criteria for convergence have been achieved.

B. The case for r values

In addition to the assumption of unique values in the previous case, it is assumed, without loss of generality, that $0 \leq x_i(0) \leq 1$. Suppose that nodes are selecting the highest r values. Hence, as is done in phase 3, in each message sent by a node, there is a list of the highest r values and their agent IDs. In this case, convergence is proven by observing the following:

(i) The system behaves like r graphs superimposed on each other. Nodes in each graph find the maximum value in their graph.

(ii) The r graphs are born from the same initial graph at the start of the communication.

(iii) In the first graph, nodes vote for only the maximum value x_{b1}. In the second graph, nodes vote for the maximum value x_{b2} with x_{b1} set to 0. In the third, nodes vote for the maximum value x_{b3} with $x_{b2} = 0$ and $x_{b1} = 0$, and so on.

(iv) Each graph may have a different sequence of adjacency matrices as the conversation progresses.

(v) Nevertheless, in each graph, only the maximum value is being voted for and is similar to the case for one value. It follows that as $k \to \infty$, $\mathbf{L}_j(k) = \mathbf{0}$ for all $j = 1, ..., r$.

Considering that the initial graph is strongly connected, the "spread" of x_b from node b to all nodes to achieve convergence can be viewed as a progression on a radial static directed acyclic graph (DAG), with node b as the root. Thus, despite the temporal nature of the graph, time analysis pertaining to static DAGs can be applied. Consequently, the speed of convergence, i.e., the number of iterations required for attaining convergence, assuming synchronism, has been shown in [37] to be bounded above by D, where D is the diameter of the graph. Thus, the time complexity is $O(D)$. In the asynchronous case, this is $O(BD)$, where B is a measure of the asynchronism [34].

It is also noteworthy that in real implementations of the algorithm, there could be $n > r$ agents with the same maximum value. Since the agent ID is stored with the selected value, it follows that until there is an explicit rule to govern how to select, there may not be convergence. This is because, while agents agree on what the maximum value is, they may not converge on the ID of the agent with this value.

Acknowledgment

The authors would like to thank the U.S. National Science Foundation (NSF) for the award of grant no. 1837359 which resulted in this research and its subsequent results.

References

[1] R. M. Lee, M. J. Assante, and T. Conway, "Analysis of the Cyber Attack on the Ukrainian Power Grid," 2016. [Online]. Available: https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf

[2] S. Nazir, S. Patel, and D. Patel, "Assessing and Augmenting SCADA Cyber Security: A Survey of Techniques," *Computers & Security*, vol. 70, pp. 436–454, 2017.

[3] P. Novák, R. Šindelář, and R. Mordinyi, "Integration Framework for Simulations and SCADA Systems," *Simulation Modelling Practice and Theory*, vol. 47, pp. 121–140, 2014.

[4] C.-W. Ten, C.-C. Liu, and G. Manimaran, "Vulnerability Assessment of Cybersecurity for SCADA Systems," *IEEE Transactions on Power Systems*, vol. 23, no. 4, pp. 1836–1846, 2008.

[5] Y. Zhang, L. Wang, Y. Xiang, and C. Ten, "Power System Reliability Evaluation with SCADA Cybersecurity Considerations," *IEEE Transactions on Smart Grid*, vol. 6, no. 4, pp. 1707–1721, 2015.

[6] Y. Yang, K. McLaughlin, S. Sezer, T. Littler, E. G. Im, B. Pranggono, and H. F. Wang, "Multiattribute SCADA-Specific Intrusion Detection System for Power Networks," *IEEE Transactions on Power Delivery*, vol. 29, no. 3, pp. 1092–1102, 2014.

[7] I. N. Fovino, A. Carcano, T. de Lacheze Murel, A. Trombetta, and M. Masera, "Modbus/DNP3 State-Based Intrusion Detection System," in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*. IEEE, 19-Apr-10 - 22-Apr-10, pp. 729–736.

[8] Y. Yang, K. McLaughlin, T. Littler, S. Sezer, B. Pranggono, and H. F. Wang, "Intrusion Detection System for IEC 60870-5-104 Based SCADA Networks," in *2013 IEEE Power & Energy Society General Meeting*. IEEE, 20-Jul-13 - 24-Jul-13, pp. 1–5.

[9] U. S. Department of Energy, "Distribution Automation: Results from the Smart Grid Investment Grants Program," September 2016. [Online]. Available: https://www.energy.gov/sites/prod/files/2016/11/f34/Distribution%20Automation%20Summary%20Report_09-29-16.pdf

[10] Z. Sun, Y. Ma, Q. Guo, and F. Sun, "Security Mechanism for Distribution Automation using EPON," in *2009 IEEE International Conference on Network Infrastructure and Digital Content*. IEEE, 05-Nov-09 - 07-Nov-09, pp. 581–585.

[11] X. Ye, J. Zhao, Y. Zhang, and F. Wen, "Quantitative Vulnerability Assessment of Cyber Security for Distribution Automation Systems," *Energies*, vol. 8, no. 6, pp. 5266–5286, 2015.

[12] Q. Dai, L. Shi, and Y. Ni, "Risk Assessment for Cyberattack in Active Distribution Systems Considering the Role of Feeder Automation," *IEEE Transactions on Power Systems*, vol. 34, no. 4, pp. 3230–3240, 2019.

[13] I. H. Lim, S. Hong, M. S. Choi, S. J. Lee, T. W. Kim, S. W. Lee, and B. N. Ha, "Security Protocols Against Cyber Attacks in the Distribution Automation System," *IEEE Transactions on Power Delivery*, vol. 25, no. 1, pp. 448–455, 2010.

[14] J. Batard, Y. Chollot, P. Pipet, L. Lamberti, and A. Gauci, "Cybersecurity for Modern Distribution Automation Grids," *CIRED - Open Access Proceedings Journal*, vol. 2017, no. 1, pp. 1002–1005, 2017.

[15] M. Parvania, G. Koutsandria, V. Muthukumary, S. Peisert, C. McParland, and A. Scaglione, "Hybrid Control Network Intrusion Detection Systems for Automated Power Distribution Systems," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 22-Jun-14 - 25-Jun-14, pp. 774–779.

[16] M. Jamei, A. Scaglione, C. Roberts, E. Stewart, S. Peisert, C. McParland, and A. McEachern, "Automated Anomaly Detection in Distribution Grids Using μ-PMU Measurements," 2016. [Online]. Available: https://arxiv.org/pdf/1610.01107.pdf

[17] Z. Li, M. Shahidehpour, A. Alabdulwahab, and A. Abusorrah, "Bilevel Model for Analyzing Coordinated Cyber-Physical Attacks on Power Systems," *IEEE Transactions on Smart Grid*, vol. 7, no. 5, pp. 2260–2272, 2016.

[18] C. Moya and J. Wang, "Developing Correlation Indices to Identify Coordinated Cyber-Attacks on Power Grids," *IET Cyber-Physical Systems: Theory Applications*, vol. 3, no. 4, pp. 178–186, 2018.

[19] C. Moya, J. Hong, and J. Wang, "Application of Correlation Indices on Intrusion Detection Systems: Protecting the Power Grid Against Coordinated Attacks," 2018. [Online]. Available: https://arxiv.org/pdf/1806.03544.pdf

[20] L. Wei, A. I. Sarwat, W. Saad, and S. Biswas, "Stochastic Games for Power Grid Protection Against Coordinated Cyber-Physical Attacks," *IEEE Transactions on Smart Grid*, vol. 9, no. 2, pp. 684–694, 2018.

[21] K. Lai, M. Illindala, and K. Subramaniam, "A Tri-level Optimization Model to Mitigate Coordinated Attacks on Electric Power Systems in a Cyber-Physical Environment," *Applied Energy*, vol. 235, pp. 204–218, 2019.

[22] S. Lakshminarayana, E. V. Belmega, and H. V. Poor, "Moving-Target Defense for Detecting Coordinated Cyber-Physical Attacks in Power Grids," in *2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, 2019, pp. 1–7.

[23] C. Sun, J. Hong, and C. Liu, "A Coordinated Cyber attack Detection System (CCADS) for Multiple Substations," in *2016 Power Systems Computation Conference (PSCC)*, 2016, pp. 1–7.

[24] J. Brown, M. Anwar, and G. Dozier, "Intrusion Detection Using a Multiple-Detector Set Artificial Immune System," in *2016 IEEE 17th International Conference on Information Reuse and Integration (IRI)*, 2016, pp. 283–286.

[25] T. Mahjabin, G. S. Y. Xiao, and W. Jiang, "A Survey of Distributed Denial-of-Service Attack, Prevention, and Mitigation Techniques," *International Journal of Distributed Sensor Networks*, vol. 13, 2017.

[26] M. E. Ajjouri, S. Benhadou, and H. Medromi, "New Collaborative Intrusion Detection Architecture Based on Multi Agent Systems," in *2015 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, 2015, pp. 1–6.

[27] M. S. Rahman, M. A. Mahmud, A. M. T. Oo, and H. R. Pota, "Multi-Agent Approach for Enhancing Security of Protection Schemes in Cyber-Physical Energy Systems," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 436–447, 2017.

[28] E. M. Amullen and L. H. Keel, "Consensus-Based Intrusion Detection for the Electric Power Grid Control System," in *2018 World Automation Congress (WAC)*, 2018, pp. 1–5.

[29] V. K. Singh, A. Ozen, and M. Govindarasu, "A Hierarchical Multi-Agent Based Anomaly Detection for Wide-Area Protection in Smart Grid," in *2018 Resilience Week (RWS)*, 2018, pp. 63–69.

[30] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang, "Machine Learning and Deep Learning Methods for Cybersecurity," *IEEE Access*, vol. 6, pp. 35 365–35 381, 2018.

[31] B. A. Akyol, J. N. Haack, S. Ciraci, B. J. Carpenter, M. Vlachopoulou, and C. W. Tews, "VOLTTRON: An Agent Execution Platform for the Electric Power System," June 2012. [Online]. Available: https://availabletechnologies.pnnl.gov/technology.asp?id=369

[32] X. Duan, J. He, P. Cheng, Y. Mo, and J. Chen, "Privacy Preserving Maximum Consensus," in *2015 54th IEEE Conference on Decision and Control (CDC)*, Dec 2015, pp. 4517–4522.

[33] B. M. Nejad, S. A. Attia, and J. Raisch, "Max-consensus in a Max-plus Algebraic Setting: The Case of Fixed Communication Topologies," in *2009 XXII International Symposium on Information, Communication and Automation Technologies*, Oct 2009, pp. 1–7.

[34] S. Giannini, D. Di Paola, A. Petitti, and A. Rizzo, "On the Convergence of the Max-Consensus Protocol with Asynchronous Updates," in *52nd IEEE Conference on Decision and Control*, Dec 2013, pp. 2605–2610.

[35] R. O. Saber and R. M. Murray, "Consensus protocols for networks of dynamic agents," in *Proceedings of the 2003 American Control Conference*, 2003, pp. 951–956.

[36] P. Biondi, "Scapy Documentation, Release 2.4.2-dev," April 2019. [Online]. Available: https://buildmedia.readthedocs.org/media/pdf/scapy/latest/scapy.pdf

[37] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Englewood Cliffs, N.J.: Prentice Hall, 1989.

Jennifer Appiah-Kubi received her BS in Electrical and Electronic Engineering at the Kwame Nkrumah University of Science and Technology (KNUST), Ghana, in 2016. She received the M.S. degree at Virginia Polytechnic Institute and State University (Virginia Tech), USA, and is currently working toward the Ph.D. degree at Virginia Tech. Her research interests include renewable energy integration and protection, and cybersecurity of power grids.

Chen-Ching Liu received his Ph.D. degree from the University of California, Berkeley. He is currently American Electric Power Professor of Electrical Engineering at Virginia Tech. Dr. Liu is also an Adjunct Full Professor of University College Dublin, Ireland. Professor Liu received an IEEE Third Millennium Medal in 2000 and IEEE Power and Energy Society Outstanding Power Engineering Educator Award in 2004. He is a Member of the U.S. National Academy of Engineering and Fellow of the IEEE.