A Trust-Management Toolkit for Smart-Grid Protection Systems

Jose E. Fadul, Kenneth M. Hopkinson, *Senior Member, IEEE*, Todd R. Andel, *Senior Member, IEEE*, and Christopher A. Sheffield

***Abstract*—This paper discusses the trust-management toolkit, which is a robust and configurable protection system augmentation, which can successfully function in the presence of an untrusted (malfunctioning) smart grid (i.e., communication-based, protection system nodes). The trust-management toolkit combines reputation-based trust with network-flow algorithms to identify and mitigate faulty smart-grid protection nodes. The toolkit assigns trust values to all protection nodes. Faulty nodes, attributed to component or communication system malfunctions (either intentional or unintentional), are assigned a lower trust value, which indicates a higher risk of failure to mitigate detected faults. The utility of the toolkit is demonstrated through simulations comparing "enhanced" backup and special protection systems to original "unenhanced" systems via an analysis of variance analysis. The results show promise for the toolkit in the smart-grid protection system.**

***Index Terms*—Protection systems, security, simulation.**

I. Introduction

THE TERM *smart grid* often refers to the use of digital equipment that employs network communication to enable greater situational awareness and cooperation for protection, control, and other power-grid mechanisms, than is traditionally possible. The goal is to improve grid efficiency and reliability [1]. The use of smart-grid technology makes it possible to improve legacy protection systems, which often mitigate detected faults with predetermined actions, by taking advantage of greater cooperation and information sharing between components. The trust-management toolkit improves the decision-making processes in communication-based protection systems in the presence of failures and disruptions attributed to malfunctions in components' sensors or communication systems.

Manuscript received May 13, 2013; revised September 10, 2013; accepted October 31, 2013. Date of publication December 03, 2013; date of current version July 21, 2014. The views expressed in this document are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government. Paper no. TPWRD-00575-2013.

J. E. Fadul and K. M. Hopkinson are with the Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright Patterson Air Force Base, OH 45433 USA (e-mail: jose.fadul@afit.edu; kenneth.hopkinson@afit.edu).

T. R. Andel is with the School of Computing, University of South Alabama, Mobile, AL 36688 USA (e-mail: tandel@southalabama.edu).

C. A. Sheffield is with the Department of Aeronautics and Astronautics, Air Force Institute of Technology, Wright Patterson Air Force Base, OH 45433 USA (e-mail: christopher.sheffield@afit.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TPWRD.2013.2289747

The proposed trust-management toolkit has several features that make it attractive. The toolkit is based entirely on software so it could be deployed and used relatively easily. As just mentioned, the toolkit is designed to augment communication-based protection schemes so that they are more robust against software and hardware failures and malfunctions. These malfunctions may be malicious, due to cyberattacks or physical tampering, or may be due to equipment failures or calibration failures occurring through other natural causes. In North America, this makes the toolkit beneficial from a North American Electric Reliability Council (NERC) critical infrastructure protection (CIP) perspective. It is also beneficial as a malfunction identification tool. Finally, the identification of malfunctioning protection nodes can be integrated into communication-based protection schemes to allow them to operate through such failures.

The envisioned target for the trust-management toolkit is transmission grids, where reliability standards and budgets are both higher.

In terms of pros and cons, the pros for the trust-management toolkit include the ability to automatically diagnose some types of hardware or software failures, the potential to detect calibration errors, the ability to enhance cybersecurity, and the ability to operate protection schemes through failures. The cons are additional complexity and the need for a communication infrastructure, though such infrastructure should already be in place since the toolkit is intended for communication-based protection schemes.

The trust-management toolkit has three major modules: 1) a trust assignment module; 2) a fault detection module; and 3) a decision module. The trust assignment module uses context-sensitive information and periodic communication to determine individual protection nodes' trust values. The fault detection module uses error signals from traditional distance relays and frequency disturbance monitoring devices to detect line and system faults, respectively. The decision module combines and analyzes current grid conditions, detected fault signals, and assigned node trust values to decide on the most reliable corrective action to mitigate detected faults.

Simulation results show promise for the trust-management toolkit in future backup and special protection systems. When all of the component nodes are trusted (functioning correctly), legacy protection systems are sufficient and the trust-management toolkit augmentation is benign—neither a benefit nor hindrance. When some nodes are untrusted (functioning incorrectly), augmented systems can detect the untrusted nodes and take appropriate mitigating actions. The usefulness of the trust-management toolkit is shown using a pedagogical backup protection system and special protection system. Computer simulations are used to compare *enhanced* backup and special protection systems to their original *unenhanced* counterparts. The main contribution of this paper is a robust and configurable trust-management toolkit protection system augmentation (also called an enhancement), which can successfully function in the presence of untrusted (malfunctioning) protection nodes.

This paper is divided into six sections and an Appendix. Section I is the introduction. Section II contains background and related work information. Section III describes the trust-management toolkit's implementation. Section IV covers the test methodology and analysis. Section V presents the experimental results and discusses their statistical significance. Section VI concludes this paper, and the Appendix presents the trust-management toolkit graph transformation pseudocode.

II. Background and Related Work

A. Background

The Energy Independence and Security Act (EISA) of 2007 [1] Title XIII sections 1301–1306, specifies smart-grid capabilities. Section 1301 identifies ten characteristics of a smart grid, whose intent is to modernize the electric power grid and enable it to safely operate closer to its peak capacity. The proposed trust-management toolkit supports seven of these ten characteristics including the increased use of digital information and controls technology to improve reliability, security, and efficiency of the electric grid, by using context-sensitive information to improve the decision-making process.

Protection systems mitigate detected faults to minimize their impact. In this paper, the trust-management toolkit enhances two of these protection systems. Backup protection systems (BPS) clear faults when primary protection fails or becomes inoperable [2]. Special protection systems (SPS) [3] monitor grid systems for known prefault conditions and take mitigating actions to prevent power-grid failures.

B. Trust-Management Toolkit-Related Work

Efforts by previous researchers [4]–[12] offer guidance on how to define and use trust in software agents. Marsh [4] formally assigned trust values to software agent interactions. Blaze *et al.* [7] coined the term *Trust Management Problem* to refer to the framework used to study security policies, credentials, and trusted relationships. Blaze *et al.* created PolicyMaker and Keynote, which utilize Pretty Good Privacy (PGP) [6] and X.509 [8] for authentication across networks.

CONFIDANT [5] and EigenTrust [9] protocols serve as inspiration for the initial incarnation of the trust-management toolkit (i.e., Simple Trust protocol [13]). Also, the preliminary work by Hopkinson *et al.* [10], Wang *et al.* [14], and Igure *et al.* [15] has shown that agents within supervisory control and data-acquisition (SCADA) systems are a viable preventative option against malfunctioning power-grid components. Furthermore, the agent-based power protection trust system developed by Borowski *et al.* [16] supports the idea of a backup protection system augmented by a trust system, which is expanded upon in our trust-management toolkit to include the status of protection components. In this paper, trust is not solely based on whether a keep-alive signal was received from a neighboring node, as it was in [16]. The graph-theoretic approach in this paper is also far more general than the one in [16]. The additional protection components' information, shared in the network, improves situational awareness and overall decision-making capabilities.

The use of redundant components to improve the reliability of their reported information is well studied. Majority voting schemes have been studied in systems where some node's information may be faulty, either accidentally or maliciously. These voting schemes are typically called Byzantine Agreement protocols [17]. In these schemes, a majority agreement can overcome faults or misinformation from others. The same basic idea is used in our trust-management toolkit. The power grid's sensors and computational components are not redundant items, but provide system information useful in determining the grid's overall state. Individual components in agreement with the system's state are considered reliable and trustworthy. One of our assumptions is that trusted components are more likely to respond correctly to control signals used to mitigate detected faults. The trust-management toolkit and its assumptions are discussed in the next section. Additional information concerning fault-tolerant systems [18] and voting algorithms [19] is available in the literature.

III. Trust-Management Toolkit

The trust-management toolkit uses network-flow techniques and reputation-based trust to improve smart-grid protection systems' resilience to intentional and unintentional protection component and communication network errors. Intentional errors are malicious acts. Unintentional errors are component failures due to normal wear and tear, manufacturing defects, etc. The trust-management toolkit consists of three modules: 1) a trust assignment module; 2) a fault detection module; and 3) a decision module. Module implementations are discussed next.

A. Trust Assignment Module

The trust assignment module assigns trust values to protection components/nodes using the *Simple Trust* algorithm. The following BPS example will help explain the mapping between assigned trust values in Table I and the power-grid components in Fig. 1. The eight square boxes in Fig. 1 represent protection nodes $(S1-S8)$ (i.e., smart BPS relays, breakers, and intelligent electronic devices (IEDs). One of these nodes is selected as the central node tasked with receiving reported sensor data. The selected node processes this data and determines trust values. Each node monitors common power-grid variables $(E_0 - E_6)$, such as line voltages, currents, impedances, etc. Each of the monitored variables is either within the node's tolerance limits or not. If a monitored variable is within tolerance, then it is assigned a bit value of "1," otherwise "0." An ordered set of 1's and 0's is used to represent the node's local status. More broadly, monitored system variables can consist of values, such as grid frequencies measured at generators and loads. The trust-management toolkit's decision module is notified when a monitored value is outside its predetermined limits. This local status is reported to a designated central node, which develops a global view of the system status based on a consensus of the reported local statuses.

TABLE I
SIMPLE TRUST ALGORITHM EXAMPLE

node	**Local status**	E_6	E_5	E_4	E_3	E_2	E_1	E_0	**Trust Value (before)**	**Trust Value (after)**
		←	**Monitored SCADA Variables**					→		
S1	123_{10}=	1	1	1	1	0	1	1	High	High
S2	123_{10}=	1	1	1	1	0	1	1	High	High
S3	109_{10}=	1	1	0	1	1	0	1	High	High
S4	7_{10}=	0	0	0	0	1	1	1	High	Low
S5	7_{10}=	0	0	0	0	1	1	1	High	Low
S6	17_{10}=	0	0	1	0	0	0	1	High	Low
S7	123_{10}=	1	1	1	1	0	1	1	High	High
S8	123_{10}=	1	1	1	1	0	1	1	High	High
Bitwise Sum		5	5	5	5	3	6	8		
Left Shift Sum		10	10	10	10	6	12	16		
Global Status		1	1	1	1	0	1	1		

Note: The "Trust Value (before)" column contains the SCADA node's trust values before the evaluation algorithm starts. The "Trust Value (after)" column contains the SCADA node's trust values after the evaluation algorithm ends.

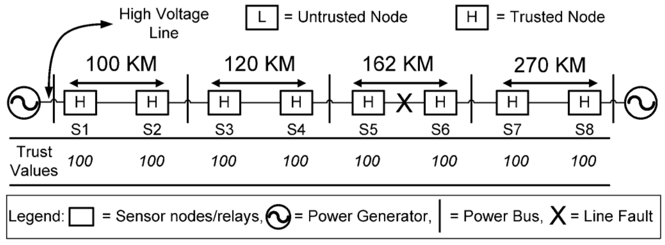

Fig. 1. Backup protection system scenario's one-line diagram with protection nodes ($S1 - S8$) corresponding to smart BPS relays, breakers, and IEDs [20].

This designated central node must be a trusted node or the trust-management toolkit may be compromised since the central node is tasked with gathering and interpreting the other nodes' status information. The simplest way to deal with this issue is to use anti-tamper hardware and a trusted platform module (TPM) in the central node's central-processing unit (CPU). This raises the cost of that node, but also greatly increases the complexity involved if adversaries attempt to compromise the proposed reputation-based trust-management toolkit.

In the central node, the global status is determined by adding all the 1's in a variable's column, multiplying the sum by 2 and then comparing the product with the number of trusted nodes contributing to the global status. If the product is greater than the number of trusted nodes contributing to the global status, then the global value for the variable is a "1," otherwise, a "0." If a majority of bits in a node's local status agrees with the global status, then it is assigned a high trust value; otherwise, it is assigned a low trust value. A low trust value indicates that the node's reported sensor data does not agree with the majority's consensus and may indicate a malfunction. A node with a low trust value may not respond correctly to commands. A high trust value indicates that the node's reported data are in agreement with the majority of reported values and suggests that all components are operating as expected. The decision module uses these trust values to determine the risk of failure associated with commanding a node to take protective action.

B. Fault Detection Module

The fault detection module uses fault detection information from the underlying protection system and feeds this information into the decision module. Generically speaking, the fault detection module monitors one or more predetermined variables for changes that indicate a condition requiring corrective action. In the backup protection system (BPS) pedagogical test cases, the variables are line impedances measured by distance relays. Special protection systems are system oriented. They exist to handle faults, which may lead to system instability rather than smaller-scope faults that are often handled by zone 1 and zone 2 relays as well as differential current protection. With this in mind, in the special protection system (SPS) test cases, the SPS system will be notified of underlying faults by existing mechanisms. Fault-location information is shared via communication network messages.

C. Decision Module

The decision module uses the previously assigned trust values to validate detected faults and determine the best response that (1) minimizes the affected area and (2) minimizes the risk of component failure when mitigating the fault. The decision module is distributed autonomous software that is capable of independent analysis and actions. The decision module's corrective actions can be determined either statically or dynamically. Static actions are predetermined and scripted (as with a truth table [21] or flowchart [20]), while dynamic actions use context-sensitive information to minimize the negative effects of a validated fault. Dynamic corrective actions entail selecting the smallest number of protection nodes with the highest probability (aka trust value) of successfully mitigating the detected fault with minimal power-grid impact. Network-flow and greedy algorithms are two dynamic approaches used to determine corrective actions.

The network-flow and greedy algorithm approaches have the same *goal or objective*, namely, to minimize a fault condition's impact. This is achieved by making the best possible *decisions* within given real-world *constraints*. The network-flow approach is used in the enhanced BPS to select the nodes closest to the detected fault with the highest probability of successfully mitigating the fault. This takes into consideration the fault's location within the power grid's topology and the trust values previously assigned to the protection nodes. The network-flow algorithm's *decision* is to select the nodes that should engage their line breakers to isolate the fault. *Constraints* include the nodes' assigned trust values and their distance from the detected fault. The fault's origin is considered to be 0 hops away from the fault. A graph representation of the faulted BPS scenario is created and solved using Dijkstra's shortest path algorithm [22]. The protection nodes traversed along the shortest path are selected to engage their line breakers and isolate the fault.

Consider the one-line diagram in Fig. 1 with a detected line fault between nodes $S5$ and $S6$. The corresponding decision graph is shown in Fig. 2. The graph-building algorithm starts by creating four artificial junction nodes; namely, the super source (S), left junction (L), right junction (R), and super sink (T) nodes. The super source and super sink nodes are the start and end nodes used in Dijkstra's shortest path algorithm. Junction nodes are used to signify the path decision nodes within the graph and are represented by circles. The left and right junction nodes indicate the left and right sides of the fault location, respectively. The graph-building algorithm, using a breadth-first search approach, progresses one hop at a time in both directions until all of the nodes have been visited or a max hop distance $(h_{\max})$ is reached. The outer while loop in algorithm 1 will terminate upon reaching the maximum hop distance $(h_{\max})$, but still provides a way to disconnect the fault edge from the rest of the power system. This is an invariant of the outer loop. The nodes are added (visited) in the following order: $S5$, $S6$, $S4$, $S7$, $S3$, $S8$, $S2$, and $S1$. The power buses between these nodes implicitly represent a complete subgraph. This implicit representation of the power buses is trivial in this simple example, but is relevant in the more complex example which follows. In this graph, only two nodes need to engage their line breakers to isolate the fault: one node on the left side of the fault and one node on the right side. The graph-edge values incident on junction nodes are zero. The graph-edge values incident on protection nodes are determined by using

$$\alpha \cdot h_j + \beta \cdot \left(\frac{1}{\tau_j}\right) \tag{1}$$

where h_j represents node j's hop distance from the fault and τ_j represents node j's trust value. Weightings α and β are set by system operators to signify the importance placed on distances from the fault and assigned trust values, respectively. The values used to generate the edge cost in Fig. 2 are shown in Table II. The shortest path between the super source and super sink nodes traverses nodes $S3$ and $S7$, indicating that their line breakers should engage to mitigate the fault.

Algorithm 1 is used to convert the power grid's topology into a graph, which is solved using Dijkstra's shortest path algorithm. This algorithm requires three pieces of information: 1) the power system topology; 2) all assigned trust values; and 3) the location of the detected line fault.

The first requirement, "knowledge of the power grid topology," is provided by system operators or a network discovery program. This information is fairly static and needs to only be updated when changes occur. The second requirement, "knowledge of all node assigned trust values," is satisfied by the trust assignment module. Trust values are broadcast to all nodes and system operators—security by obscurity is not used here. This ensures that system operators are aware of the protection component trustworthiness and communication capabilities. The third requirement, "knowledge of the location of the detected line fault," is provided by the fault detection unit. Pseudocode for the graph generation algorithm appears in the Appendix at the end of this paper. A flowchart of the pseudocode is given in Fig. 3(a) and (b). This flowchart breaks a recursive version of the code down into simple steps and gives both code and commented explanations for the steps. The full code in the Appendix differs in that it is not recursive. Also, the code in the Appendix can handle junctions with unlimited numbers of branches whereas the flowchart version is limited to junctions with right and left branches. Nonetheless, the flowchart in Fig. 3(a) and (b) can be easier to understand than the full version in the Appendix and is similar in spirit to the full pseudocode in the Appendix.

This graph-building algorithm can be applied to more complex topologies, as in Fig. 4. If a fault is detected in the Fig. 4 one-line diagram between nodes $S11$ and $S12$, then the Fig. 5 decision graph is generated. Junction nodes $J1$ through $J5$ represent decision points where multiple paths are possible.

The objective function being minimized by Dijkstra's shortest path algorithm is

$$\sum_{j=1}^{N} \left\{ \left[\alpha \cdot h_j + \beta \cdot \left(\frac{1}{\tau_j}\right) \right] \cdot \zeta_j \right\} \tag{2}$$

where N is the number of protection nodes and ζ_j is a selection variable equal to "1" when node j is on the shortest path and "0" otherwise. Minimizing (2) tends to minimize the risk of failure to mitigate the fault and the size of the affected area.

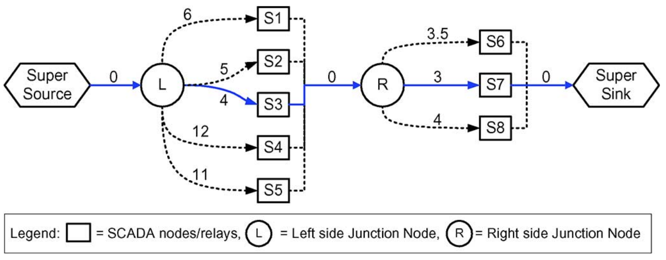
Super Source
0
L
6
5
4
12
11
S1
S2
S3
S4
S5
0
R
3.5
3
4
S6
S7
S8
0
Super Sink
Legend: ☐ = SCADA nodes/relays, (L) = Left side Junction Node, (R) = Right side Junction Node

Fig. 2. Decision module's generated graph for power-grid diagram Fig. 1 [20].

TABLE II
INCIDENT NODE-EDGE VALUES

Node ID	j	α	β	h_j	τ_j	Cost
S1	1	1	100	5	100	6
S2	2	1	100	4	100	5
S3	3	1	100	3	100	4
S4	4	1	100	2	10	12
S5	5	1	100	1	10	11
S6	6	1	100	1	40	3.5
S7	7	1	100	2	100	3
S8	8	1	100	3	100	4

Trust Management System Transformation Pseudocode(N, E, F)

This algorithm uses a SCADA power grid connectivity graph (N, E) and a fault location edge (F) to construct a graph problem, solvable by Dijkstra's shortest path algorithm.

```
1
1. Procedure SCADA_TMS_Transformation
Inputs: $N$ is the set of SCADA nodes within the SCADA Power Grid
$E$ is the set of SCADA edges within the SCADA Power Grid
$F$ is the SCADA edge experiencing a Fault, i.e. $F \in E$.
Note: Each edge is represented by a set of two nodes, i.e.,
$e \in E, = \{left, right\}$,
where $left, right \in N$ and accessed by $e.left = left$ and
$e.right = right$.

2
2. Begin
// Initialization
3. $\hat{N} \leftarrow \emptyset$, $\hat{E} \leftarrow \emptyset$ // clear return variables--Node and Edge sets
4. $Left_{root} \leftarrow F.left$, $Right_{root} \leftarrow F.right$ // get fault end nodes, both
                                                // left and right
5. $E \leftarrow E - F$ // remove fault edge from given set of Edges
6. $N \leftarrow N - Left_{root}$ // remove left root node from given set of Nodes
7. $N \leftarrow N - Right_{root}$ // remove right root node from given set of Nodes
8. $h \leftarrow 1$, // set hop count value to 1
9. $h_{max} \leftarrow 10$ // set max hop value to 10
10. $\alpha \leftarrow 1$ // set hop distance weighting factor to 1
11. $\beta \leftarrow 100$ // set trust weighting factor to 100

3
// Add Super Source, Left Junction, Right Junction and Super Sink
// Nodes to return variables, $N$ and $E$
12. $\hat{N} \leftarrow \hat{N} \cup \{S\} \cup \{L\} \cup \{R\} \cup \{T\}$
13. $\hat{E} \leftarrow \hat{E} \cup \{S,, cost \leftarrow 0\}$
14. $\hat{E} \leftarrow \hat{E} \cup \{L, Left_{root}, cost \leftarrow \alpha * h + \beta *$
    $(1/Left_{root}.trust)\} \cup \{Left_{root}, R, cost \leftarrow 0\}$

4
// Call recursive helper function for left side nodes
15. Build_Graph($N, E, \hat{N}, \hat{E}, Left_{root}, L, R, h, h_{max}$)
16. $\hat{E} \leftarrow \hat{E} \cup \{R, Right_{root}, cost \leftarrow \alpha * h + \beta * (1/Right_{root}.trust)\} \cup$
$\{Right_{root},, cost \leftarrow 0\}$

5
// Call recursive helper function for right side nodes
17. Build_Graph$N,, \hat{N}, \hat{E}$ $Right_{root}, R, T, h, h_{max}$)
18. Return $\hat{N}$, $\hat{E}$
19. End procedure SCADA_TMS_Transformation
```

(a)

Build_Graph($N, E, \hat{N}, \hat{E}, root, j, \Omega$, h, h_{max}) Pseudocode

This helper algorithm builds a graph based on the power grid topology

```
1
1. Procedure Build_Graph
// Inputs: $N$ is the set of SCADA nodes within the SCADA Power
// Grid, see Figure 19
// $E$ is the set of SCADA edges within the SCADA Power Grid, see
// Figure 19
// $\hat{N}$ is the set of network flow problem nodes
// $\hat{E}$ is the set of network flow problem edges
// $root$ is the starting node in $N$
// $j$ is the source node in $\hat{N}$
// $\Omega$ is the end node in $\hat{N}$
// $h_{max}$ is the maximum allowed hop distance from the fault
// Note: all above variables are passed by reference
// $h$ is the hop distance from the fault

2
2. Begin
// Check Exit conditions
3. If ($E == \emptyset$) or ($N == \emptyset$) or ($h \geq h_{max}$)
4. Return
5. End If

3
// Initialization
6. $temp\_edges \leftarrow \emptyset$, $temp\_nodes \leftarrow \emptyset$, $h \leftarrow h+1$, $\alpha \leftarrow 1$, $\beta \leftarrow 100$

4
// find all nodes adjacent to $root$ in $N$
7. For each $e$ in $E$
8. If ($e.left == root$) and ($e.right \in N$)
9. $temp\_edges \leftarrow temp\_edges \cup e$
10. $temp\_nodes \leftarrow temp\_nodes \cup e.right$
11. End If
12. If ($e.right == root$) and ($e.left \in N$)
13. $temp\_edges \leftarrow temp\_edges \cup e$
14. $temp\_nodes \leftarrow temp\_nodes \cup e.left$
15. End If

5
// Update the given nodes and edges
17. $E \leftarrow E - temp\_edges$
18. $N \leftarrow N - temp\_nodes$
19. If ($temp\_edges == \emptyset$) or ($temp\_nodes == \emptyset$)
20. Return
21. End If
22. If ($|temp\_nodes| == 1$)
23. $\hat{N} \leftarrow \hat{N} \cup temp\_nodes$
24. $\hat{E} \leftarrow \hat{E} \cup \{j, temp\_nodes.element, cost \leftarrow$
                 $\alpha * h + \beta * (1/temp\_nodes.element.trust)\}$
25. $\hat{E} \leftarrow \hat{E} \cup \{temp\_nodes.element, \Omega, cost \leftarrow 0\}$
26. Build_Graph($N, E, \hat{N}, \hat{E}, temp\_nodes.element, j, \Omega, h, h_{max}$)
27. Return
28. End If

6
// spit point found requiring addition virtual junction nodes
29. $Junction_{in} \leftarrow new\ virtual\ Junction\ node$
30. $\hat{N} \leftarrow \hat{N} \cup \{junction_{in}\}$
31. $\hat{E} \leftarrow \hat{E} \cup \{\{j, junction_{in}, cost \leftarrow 0\}\}$
32. For each $node$ in $Temp\_nodes$
33. $junction_{out} \leftarrow new\ virtual\ junction\ node$
34. $\hat{N} \leftarrow \hat{N} \cup \{junction_{out}\}$
35. Build_Graph($N, E, \hat{N}, \hat{E}, node, junction_{in}, junction_{out}, h, h_{max}$)
36. $junction_{in} \leftarrow junction_{out}$
37. End For each
38. $\hat{E} \leftarrow \hat{E} \cup \{junction_{out}, \Omega, cost \leftarrow 0\}$
39. Return
40. End procedure Build_Graph
```

(b)

Fig. 3. (a). Pseudocode for a recursive version of the algorithm in the Appendix. This code is broken down into blocks by function and includes comments. Please note that this version of the code can only handle left and right branches at junctions whereas the Appendix can handle more complex cases. (b). The helper routine for the pseudocode in Fig. 3(a).

In Fig. 1, the low-trust values assigned to $S4$, $S5$, and $S6$ indicate that their reported data disagree with the majority of received data and indicate possible malfunctions or communication failures. This raises questions regarding their ability to re-

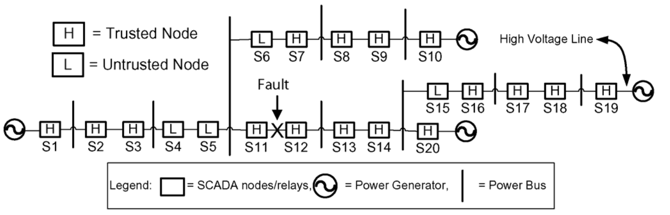
H = Trusted Node
L = Untrusted Node
S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12 S13 S14 S15 S16 S17 S18 S19 S20
Fault
High Voltage Line
Legend: = SCADA nodes/relays, = Power Generator, = Power Bus

Fig. 4. Representative power-grid topology with assigned trust values with nodes ($S1S20$) corresponding to smart BPS relays and breakers.

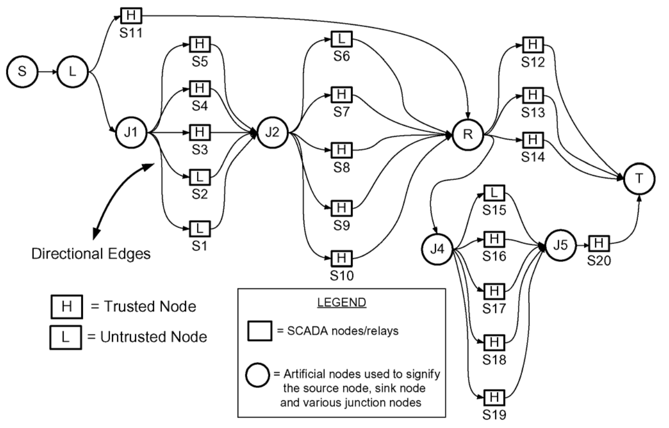
S
L
J1
J2
R
J4
J5
T
S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12 S13 S14 S15 S16 S17 S18 S19 S20
Directional Edges
H = Trusted Node
L = Untrusted Node
LEGEND
= SCADA nodes/relays
= Artificial nodes used to signify the source node, sink node and various junction nodes

Fig. 5. Graph for a detected fault between nodes $\langle S11, S12\rangle$ in Fig. 4.

ceive commands and operate correctly. This risk of failure to mitigate a fault is one factor considered. Another factor is increased outage size. The trust-management toolkit attempts to determine the most reliable solution with the least grid impact. Nodes $S3$ and $S7$ are selected to mitigate the fault in Fig. 1 based on their trust values and proximity to the fault.

The greedy algorithm is another dynamic approach used in the enhanced SPS to select trusted protection nodes that participate in demand response to load shed a calculated amount and prevent a power outage. This takes into consideration the nodes' assigned trust values and the amount of power being shed. The *decision* being made is the selection of nodes commanded to load shed a specific amount of power. The *constraints* include selecting trusted nodes before selecting untrusted nodes and selecting nodes participating in demand response before selecting nodes that are not participating in demand response. The *goal or objective* is to prevent a power outage by commanding nodes to load shed a calculated amount resulting in a system frequency that remains above 58.8 Hz [10].

A sorting algorithm is used to determine the order in which the protection system nodes are selected for load shedding. This sorting algorithm uses demand response data in conjunction with assigned trust values, node types, and load values to sort the protection nodes. Nodes are sorted by an order of precedence first sorted by *Type*, then *Trust Value*, *Demand Response* participation, and by *Load (MW)*. Trusted load nodes are chosen before untrustworthy nodes. The Trust Management Toolkit assumptions are as follows.

1) Each node's sensor is able to collect and process all of their sensor data within the simulation time step of 2 ms.
2) All malfunctions are attributed to a node's faulty sensors, processing components, transmission medium, or communications equipment. If one aspect of a node fails, then all internal functions become equally suspect.
3) All functioning nodes can communicate with each other.
4) The number of untrusted nodes is always a minority.
5) The designated central node (used to determine trust values) must be a trusted node.
6) Given 2) above, faulty nodes cannot mitigate line faults.

One point briefly discussed so far is the source of power-grid topology/connectivity and fault-location information.

There are several sources from which power-grid topology/connectivity information might be derived from. The simplest source is to hard code such information into each device upon its installation. This is less than ideal since updates are hard to propagate throughout the power network. Creating such information per device can also be tedious and error prone if done manually. A better method is to automate the topology information and update it over time. This could perhaps be derived from the SCADA system. Since the target of the trust-management toolkit is communication-based protection systems, the data could also be derived from the underlying communication network using tools like nmap [23], assuming that the communication topology matches the power topology. As wide-area measurement systems and their associated networks become more prevalent, this will be an increasingly realistic option.

Since the trust-management toolkit does not have a facility for detecting electric power faults, it would rely on the underlying protection system for information regarding a fault's location. Most short-circuit faults, for example, will be detected by underlying relays in zones 1 and 2 and by differential current protection schemes.

In the case of connectivity/topology and fault information, the information can be collected and disseminated through the underlying communication-based protection system as required by using the network that must necessarily be in place for regular protection system operation.

D. Trust-Management Toolkit Test Cases

The utility of the proposed trust-management toolkit is demonstrated by computer simulations of enhanced backup and special protection systems using power test cases. The protection systems are enhanced as follows:

1) Enhanced Backup Protection System (BPS): The communication-based backup protection system (BPS), as described in [10], uses a communication network to coordinate relays to respond to a fault. This BPS has been enhanced by incorporating the proposed trust-management toolkit into its operating logic. The assignment module in the trust-management toolkit uses monitored power-grid information in a reputation-based trust algorithm called Simple Trust. The average of the ten most recent trust values from Simple Trust is used to establish and assign individual node trust values. The fault detection module uses the traditional distance relay mechanism to detect line faults. The decision module minimizes (2) to determine which breakers to engage. The idea is to allow the BPS to incorporate how trustworthy a relay is, based on reputation-based trust values, when determining how to coordinate the response to a fault.

2) Enhanced Special Protection System (SPS): A communication-based special protection system, previously introduced in [10], is also enhanced with the trust-management toolkit in this paper. The trust-management toolkit assignment module monitors the system frequency and uses its tolerance results in a reputation-based trust algorithm called Simple Trust to establish and assign trust values. The fault detection module uses the traditional frequency disturbance mechanism to detect changes in system conditions indicative of an imminent underfrequency fault condition. The decision module uses a greedy algorithm to determine the nodes to select for load shedding. Once again, the goal is to augment the capabilities of the communication-based special protection system so that it can use trustworthiness, based on reputation-based trust ratings, when determining the best response to a fault.

IV. Methodology

A. Testing Environment

Protection test simulations utilized the electric power and communication synchronizing simulator (EPOCHS) [10]. EPOCHS federates, or combines, Network Simulator 2 (NS2) [24], Power Systems Computer Aided Design (PSCAD) [25], and the Power System Simulation for Engineering (PSS/E) [26]. NS2 is an open-source network simulator. PSCAD and PSSE are commercial electromagnetic and electromechanical power transient simulators, respectively. EPOCHS coordinates and controls the interactions between NS2 and the power simulators. Raw simulation data are analyzed with the open-source R statistical package [27]. An analysis of variance (ANOVA) comparison of confidence intervals [28] is used to determine the statistical significance of the simulation results. EPOCHS facilitates realistic simulations of smart-grid systems, which consist of electrical and network communication elements.

B. Enhanced Backup Protection System (BPS) Test Scenario

BPS are tasked with clearing faults when primary protection fails to act [2]. The BPS test scenario in Fig. 1 consists of two generators supplying power across five buses in a 400-kV power system. The Bergeron line model is used in our test case. Distances are given in Fig. 1. The trust assignment modules receive data readings from all relay nodes and determine that nodes $S4$, $S5$, and $S6$ are untrusted with trust values of 10%, 10%, and 40%, respectively. The remaining nodes are fully trusted. The fault detection modules sense a line fault between nodes $S5$ and $S6$. The detected fault's location is shared among the nodes via network broadcast messages. The decision modules receive the fault messages and use the assigned trust values to transform the one-line diagram in Fig. 1 into the graph in Fig. 2. The cost assigned to the edges entering the nodes in Fig. 2 is determined by using (1), and the remaining edges are assigned a cost of zero. The lower trust values in Fig. 1 correspond to the higher edge costs in Fig. 2 and Table II. The decision module uses Dijkstra's shortest path algorithm on the generated graph to determine the shortest path between the super source and super sink nodes. This path traverses nodes $S3$ and $S7$, indicating that their breakers should be engaged to isolate the fault. As a result, the decision modules at nodes $S1$, $S2$, $S3$, and $S8$ send trip messages to node $S7$. The decision module at $S7$ receives and processes the trip messages, which causes it to trip its line breakers. In addition, the decision modules at nodes $S1$, $S2$, $S7$, and $S8$ send trip messages to node $S3$. The decision module at $S3$ also receives and processes the trip messages, which causes it to trip its line breakers to isolate the fault.

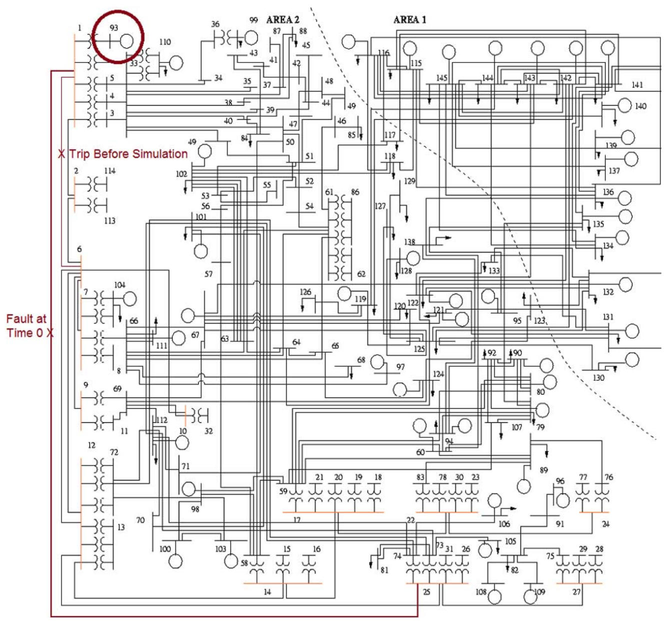

Fig. 6. Modified IEEE 50-generator/145-bus power test case. The 1–6 bus trips before the simulation begins. The 1–25 bus faults at time 0. Generator 93 is rejected at time 0.10 s. (This figure is a modified version of the one provided to us by Vijay Vittal.)

C. Enhanced Special Protection System (SPS) Test Scenario

Special protection systems (SPS) [3] monitor one or more power systems for error conditions and take predetermined actions to pre-emptively correct such errors. The SPS test scenarios monitor the system frequency for disturbances that are indicative of an imminent fault and attempt to prevent the fault by generation rejection and load shedding. A modified version of the IEEE 50-generator/145-bus power test case [29] is used within PSS/E to demonstrate the enhanced SPS benefits. This test case was modified to better represent an electric power-grid system requiring an SPS. This system is depicted in Fig. 6. A detailed summary of the modifications is available in [10].

In this modified test case, a series of events occurs, causing an SPS condition. Two intertie power lines are lost, causing the system to become transiently unstable and necessitating generation rejection. Generator 93 was preselected by system operators for generation rejection and commanded to trip. This counteracts the transient instability, but causes an unacceptable decrease in frequency. The goal is to keep the frequency above 58.8 Hz [10]. The enhanced SPS system uses the disturbance size to estimate the load shed amount required to maintain the power grid's frequency above 58.8 Hz. This load shed amount is levied on selected loads. If the selected load shedding nodes are untrusted and refuse to shed load, the SPS will fail to maintain

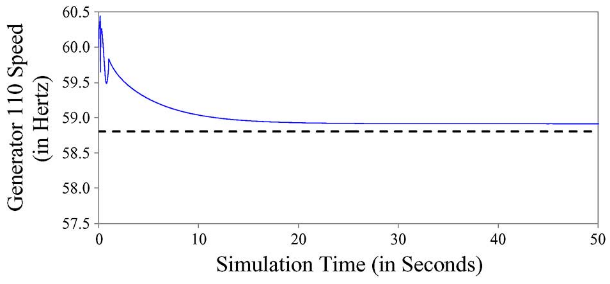

Fig. 7. Enhanced SPS is able to keep the frequency above 58.8 Hz.

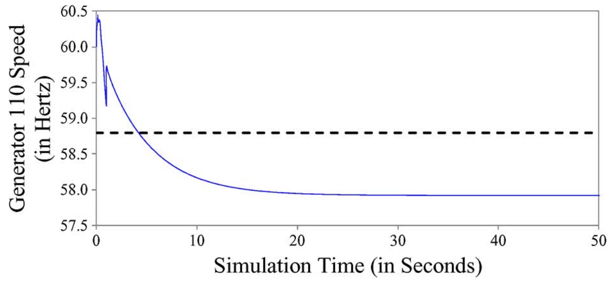

Fig. 8. Unenhanced SPS is unable to keep the frequency above 58.8 Hz.

a system frequency above 58.8 Hz. This underlines the importance in the decision module's selection of trusted nodes for load shedding. Recall that trusted nodes have a higher probability of successfully completing an assigned task than untrusted nodes.

The trust assignment module uses the frequency information provided by the generators and key loads to determine their individual trust values. The fault detection module senses the frequency disturbance, created by losing the generator at node 93, and estimates the load shedding amount required to maintain the system's frequency above 58.8 Hz. The decision module selects nodes for load shedding based on node type, their assigned trust values, demand response data, the load, the load available for shedding, and the load that must be shed. The decision module sends each selected node a load shed command with the amount identified. The trusted nodes shed their assigned amounts, which keeps the frequency above 58.8 Hz and helps avert a probable power outage.

D. Trust-Management Toolkit Simulation Test Procedures

The BPS scenarios conducted in PSCAD are fully covered in [20] and [21] with the results summarized in this paper. The SPS test case conducted in PSS/E is a 145-bus IEEE test case, which is described in this section.

Thirty six of NS2's predefined "good" random number seeds were used for each test case configuration. The data collected are the minimum power-grid system frequency. This frequency value may also be considered as the system's steady-state frequency or the asymptotic frequency of the run, as depicted in Figs. 7 and 8. The frequency data from the "unenhanced SPS test case with 5 untrusted nodes" were used to determine the data's normality.

The normality of the sample data was confirmed by the Shapiro–Wilk normality test [30], which results in a p—*value* of 0.5598 and a W value of 0.9745. The analysis of variance

```
Analysis of Variance Table

Response: Frequency
                  Df  Sum Sq  Mean Sq  F value      Pr(>F)
Treatment          1  5.3865   5.3865  1292.86   < 2.2e-16  ***
Levels             2  0.8682   0.4341   104.19   < 2.2e-16  ***
Treatment:Levels   2  0.9637   0.4818   115.65   < 2.2e-16  ***
Residuals        210  0.8749   0.0042
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

ANOVA numerical results showing evidence of a statistical difference between enhanced and unenhanced SPS systems.

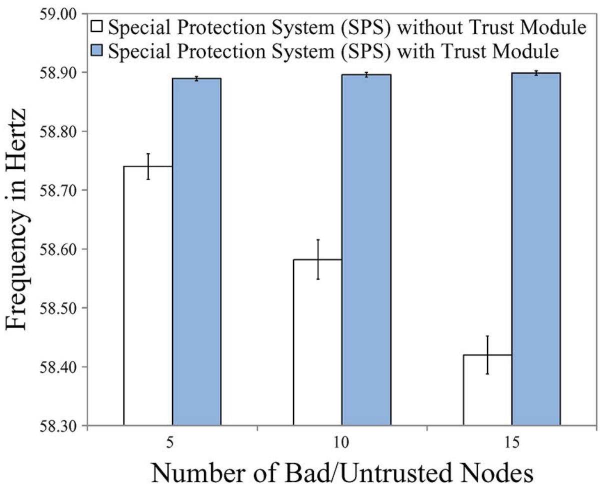

Fig. 9. Comparison of test treatments with 5, 10, and 15 untrusted nodes.

test for normality by Shapiro and Wilk confirmed that the collected sample data were drawn from a normally distributed population with a 95% confidence interval. This result, coupled with the sample size of 36 data points per treatment, justifies our use of the z-statistics in our statistical analysis of the simulation results that were generated in our tests.

The SPS simulations have two factor: 1) number of untrusted nodes and 2) whether the SPS is enhanced with the trust-management toolkit. The three levels of untrusted nodes are 5, 10, and 15. A random number generator is used to select the untrusted nodes. Hence, the total number of experiments is six (i.e., $3 * 2$). Each experiment is run 36 times to obtain the sample frequency data. The samples' mean steady-state frequency and confidence interval of each experiment are calculated and plotted in Fig. 9. Clearly, the statistical results depicted in Fig. 9 show a statistically significant difference between the enhanced and unenhanced SPS systems. The graphical results in Fig. 8 are referred to by [31] and [32] as an ANOVA, where the variance about the mean values is represented by a 95% confidence interval. The nonoverlapping confidence intervals illustrate a statistically significant difference between the enhanced and unenhanced SPS systems. The R statistical package ANOVA numerical calculation results are provided in the Analysis of Variance Table above as additional evidence of a statistical difference between the two treatments (enhanced and unenhanced). The p-value (aka Pr (>F)) is less than 2.2×10^{-16}, which is smaller than the alpha value of 0.05 associated with a confidence interval of 95%. This small p-value is convincing evidence of a statistical difference between the two treatments.

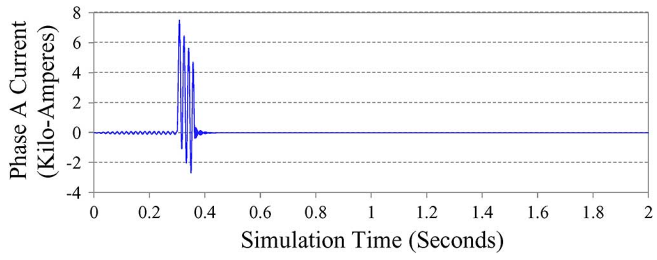

Fig. 10. Trust-management toolkit-enhanced backup protection system isolates the faulty line quickly—in ~0.022 s.

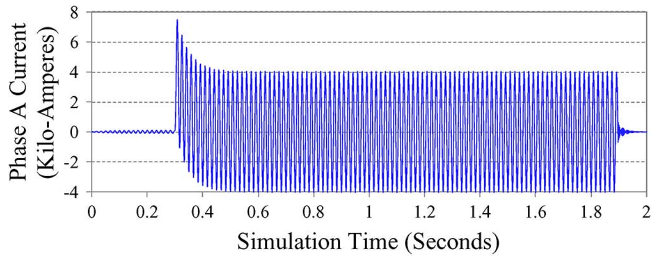

Fig. 11. Traditional unenhanced backup protection system isolates the faulty line in ~1.5 s.

V. Simulation Results

Results from the backup protection system scenarios conducted in PSCAD, fully covered in [20] and [21], are summarized here. Results from SPS simulations are also presented here.

A. BPS Scenario Results

The enhanced BPS was able to detect, initiate corrective action, and isolate a faulty power line between nodes $S5$ and $S6$ by commanding the nodes located at $S3$ and $S7$ to engage their breakers. Cooperation through communication yields the improved BPS response time in Fig. 10. The unenhanced BPS can also correct the problem, but must wait a predetermined amount of time for $S5$ and $S6$ to respond to the problem. If $S5$ and $S6$ fail to respond, then the following nodes ($S4$ and $S7$) are tasked with engaging their line breakers to correct the problem. If $S7$ succeeds in engaging its line breaker but $S4$ fails to engage its line breaker, then $S3$ is tasked with engaging its line breaker—as a backup to $S4$. In this unenhanced BPS scenario, it takes approximately 1.5 s for the system to isolate the faulty line, see Fig. 11. These results from this example indicate a 99% improvement in backup protection system response time.

B. SPS Scenario Results

The enhanced SPS is able to keep the system's frequency above 58.8 Hz across all three levels of untrusted nodes; namely, the 5, 10, or 15 untrusted nodes levels. In the simulation runs, untrusted nodes are malfunctioning nodes, meaning that they do not shed load when commanded to do so. The unenhanced SPS is not able to keep the power grid's frequency above 58.8 Hz, while the enhanced SPS does keep the frequency above 58.8 Hz. The simulation results are statistically supported by Fig. 9. This figure shows that, on average, the enhanced SPS was able to maintain the power grid's frequency above 58.8 Hz under all three levels of untrusted nodes, while the unenhanced SPS system could not under any of the three levels of untrusted nodes.

As expected, the number of untrusted nodes has no effect on the enhanced SPS system. The unenhanced SPS system is adversely affected by the increasing number of untrusted nodes. In particular, as the number of untrusted nodes increases, so does the detrimental effect on the unenhanced SPS system. At a 95% confidence interval, it is clear that across the three levels of untrusted nodes, the enhanced SPS system has a higher minimum frequency compared to the unenhanced SPS system. This highlights the utility of the trust-management toolkit to future smart-grid protection systems.

VI. Conclusion

This paper introduces a robust and configurable trust-management toolkit, which allows protection systems to successfully function in the presence of untrusted (malfunctioning) components. The toolkit combines a reputation-based trust with network-flow algorithms to identify faulty protection and/or communication components. Simulation results show promise for the proposed trust-management toolkit. Such tools have the potential to increase the robustness of smart-grid protection systems, while lowering the risk of faulty component-related power outages.

Appendix

This appendix gives pseudocode for Algorithm 1, the Trust Management Toolkit Transformation algorithm, used in this paper.

Algorithm 1 ***Trust Management Toolkit Transformation Pseudocode (N, E, F)***

```
1.  Global Variables: $h_{max} \leftarrow 10, \alpha \leftarrow 1, \beta \leftarrow 1$
        // hmax is maximum hop distance
2.  Global functions: double Cost(hops, node)
        $\{return\ \alpha \cdot \text{hops} + (\beta/\text{node.trust}); \}$
3.  Require: $N \neq \emptyset \wedge E \neq \emptyset \wedge F \neq \emptyset \wedge F \in E$
      // where: $N$ is the set of SCADA nodes like in Fig. 1
      //    $E$ is the set of SCADA edges like in Fig. 1
      //    $F$ is the SCADA edge experiencing a Fault
      // Note: edges are an ordered set of SCADA nodes, i.e.,
      // $edge = (x, y)$ where $x \neq y \wedge x < y \wedge x, y \in N$ and
      // $edge.left = x$ (the left projection of the edge pair) and
      // $edge.right = y$ (the right projection of the edge pair).
4.  Return Variables: $\widehat{N}$ and $\widehat{E}$
      // where: $\widehat{N}$ is a set of nodes and $\widehat{E}$ is a set of edges.
      // Note: the returned edges have an added tuple value
      of cost,
        // i.e., $edge \in \widehat{E}$ such that $edge = (x, y, cost)$ where $x \neq y \wedge$
        // $x < y \wedge x, y \in \widehat{N} \wedge cost \in \Re$ and $edge.left = x$ and
        // $edge.right = y$ and $edge.cost = cost$
5.  Start Procedure: Trust Management Toolkit Transformation
// Initialization
6.  $\widehat{N} \leftarrow \emptyset$    // a Set Data Structure
7.  $\widehat{E} \leftarrow \emptyset$    // a Set Data Structure
8.  $Q \leftarrow \emptyset$    // a Queue Data Structure
9.  $temp\_nodes \leftarrow \emptyset$    // a Queue Data Sructure
10.  $h \leftarrow 1$    // an Integer Variable
11.  $count \leftarrow 1$    // an Integer Variable
// Update Given sets of Nodes and Edges by removing
// $F$ from set $E$ and the end nodes of $F$ from set $N$.
12.  $E \leftarrow E - F$
13.  $N \leftarrow N - F.left$
14.  $N \leftarrow N - F.right$
// Add super source $(S)$, left junction $(L)$, right junction $(R)$,
// super sink $(T)$, left fault and right fault nodes to return variables,
// $\widehat{N}$ and $\widehat{E}$ like in Fig. 2
15.  $\widehat{N} \leftarrow \{S\} \cup \{L\} \cup \{R\} \cup \{T\}$
16.  $\widehat{E} \leftarrow \{(S, L, 0)\}$
// Add to Q; {previous_node, current_node, next_node,
// Number_of_hops_to_current_node
17.  $Q.enqueue((L, F.left, R, h))$
18.  $\widehat{E} \leftarrow \widehat{E} \cup \{(L, F.left, Cost(h, F.left))\}$
19.  $\widehat{E} \leftarrow \widehat{E} \cup \{(F.left, R, 0)\}$
20.  $\widehat{N} \leftarrow \widehat{N} \cup \{F.left\}$
21.  $Q.enqueue((L, F.right, R, h))$
22.  $\widehat{E} \leftarrow \widehat{E} \cup \{(L, F.right, Cost(h, F.right))\}$
23.  $\widehat{E} \leftarrow \widehat{E} \cup \{(F.right, R, 0)\}$
24.  $\widehat{N} \leftarrow \widehat{N} \cup \{F.right\}$
// Start double nested while loops
25.  while $(Q.size(\ ) \neq 0 \wedge E \neq \emptyset \wedge N \neq \emptyset \wedge Q.firstItem(\ ).hops \leq h_{max})$ do
26.      for all $e \in E$ do
27.          if $(e.left == Q.firstItem(\ ).current\_node \wedge e.right \in N)$ then
28.              $E \leftarrow E - e$
29.              $temp\_nodes.enqueue(e.right)$
30.          end if
31.          if $(e.right == Q.firstItem(\ ).current\_node \wedge e.left \in N)$ then
32.              $E \leftarrow E - e$
33.              $temp\_nodes.enqueue(e.left)$
34.          end if
35.      end for
36.      if $(temp\_nodes.size(\ ) == 1)$ then
37.          $Q.enqueue((Q.firstItem(\ ).previous\_node, temp\_nodes.firstItem(\ ), Q.firstItem(\ ).next\_node, Q.firstItem(\ ).hops + 1))$
38.          $\widehat{E} \leftarrow \widehat{E} \cup \{(Q.firstItem(\ ).previous\_node, temp\_nodes.firstItem(\ ), Cost(Q.firstItem(\ ).hops + 1, temp\_nodes.firstItem(\ )))\}$
39.          $\widehat{E} \leftarrow \widehat{E} \cup \{(temp\_nodes.firstItem(\ ), Q.firstItem(\ ).next\_node, 0)\}$
40.          $\widehat{N} \leftarrow \widehat{N} \cup \{temp\_nodes.firstItem(\ )\}$
41.          $N \leftarrow N - \{temp\_nodes.firstItem(\ )\}$
42.          $temp\_nodes.dequeue(\ )$
43.      end if // Here: $temp\_nodes.size(\ ) == 0$
44.      if $(temp\_nodes.size(\ ) \geq 1)$ then
45.          $artificial\_JR \leftarrow new\_node(\text{'J'} + count)$
46.          $\widehat{E} \leftarrow \widehat{E} \cup \{(Q.firstItem(\ ).previous\_node, artificial\_JR, 0)\}$
47.          $\widehat{N} \leftarrow \widehat{N} \cup \{artificial\_JR\}$
48.          while $(temp\_nodes.size(\ ) \neq 0)$ do
49.              $count \leftarrow count + 1$
50.              $artificial\_JL \leftarrow artificial\_JR$
51.              $artificial\_JR \leftarrow new\_node(\text{'J'} + count)$
52.              if $(temp\_nodes.size(\ ) == 1)$ then
53.                  $Q.enqueue((artificial\_JL, temp\_nodes.firstItem(\ ), Q.firstItem(\ ).next\_node, Q.firstItem(\ ).hops + 1))$
54.                  $\widehat{E} \leftarrow \widehat{E} \cup \{(artificial\_JL, temp\_nodes.firstItem(\ ), Cost(Q.firstItem(\ ).hops + 1, temp\_nodes.firstItem(\ )))\}$
```

```
55.             $\widehat{E} \leftarrow \widehat{E} \cup \{(temp\_nodes.firstItem(\,), Q.firstItem(\,).next\_node, 0)\}$
56.             $\widehat{N} \leftarrow \widehat{N} \cup \{temp\_nodes.firstItem(\,)\}$
57.         else // $temp\_nodes.size(\,) > 1$
58.             $Q.enqueue((artificial\_JL, temp\_nodes.firstItem(\,), artificial\_JR, Q.firstItem(\,).hops + 1))$
59.             $\widehat{E} \leftarrow \widehat{E} \cup \{(artificial\_JL, temp\_nodes.firstItem(\,), Cost(Q.firstItem(\,).hops+1, temp\_nodes.firstItem(\,)))\}$
60.             $\widehat{E} \leftarrow \widehat{E} \cup \{(temp\_nodes.firstItem(\,), artificial\_JR, 0)\}$
61.             $\widehat{N} \leftarrow \widehat{N} \cup \{temp\_nodes.firstItem(\,)\}$
62.             $\widehat{N} \leftarrow \widehat{N} \cup \{artificial\_JR\}$
63.         end if
64.         $N \leftarrow N - \{temp\_nodes.firstItem(\,)\}$
65.         $temp\_nodes.dequeue(\,)$
66.       end while
67.     end if // Here: $temp\_nodes.size(\,) == 0$
68.     $Q.dequeue(\,)$ //remove first item from queue
69.   end while
70.   return $\widehat{N}, \widehat{E}$
71. End procedure
```

Note: This algorithm uses a power-grid connectivity graph (N, E) and a fault-location edge (F) to construct a graph $(\widehat{N}, \widehat{E})$, solvable by Dijkstra's shortest path algorithm.

REFERENCES

[1] H.R.6-110th Congress, Energy Independence and Security Act (EISA) of 2007 GovTrack.us (database of federal legislation), 2007.

[2] *IEEE 100: The Authoritative Dictionary of IEEE Standards Terms*, 7th ed. New York: IEEE, 2000.

[3] P. M. Anderson and B. K. LeReverend, "Industry experience with special protection schemes," *IEEE Trans. Power Syst.*, vol. 11, no. 3, pp. 1166–1179, Aug. 1996.

[4] S. P. Marsh, "Formalising trust as a computational concept," Ph.D. dissertation, Dept. Comput. Sci. Math., Univ. of Stirling, Stirling, Scotland, U.K., 1994.

[5] S. Buchegger and J.-Y. L. Boudec, "Performance analysis of the CONFIDANT protocol," in *Proc. 3rd ACM Int. Symp. Mobile ad hoc Netw. Comput.*, New York, USA, 2002, pp. 226–236.

[6] P. R. Zimmermann, *The Official PGP User's Guide*. Cambridge, MA, USA: MIT Press, 1995.

[7] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *Proc. IEEE Symp. Security Privacy*, 1996, pp. 164–173.

[8] R. Housley, W. Polk, W. Ford, and D. Solo, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. Fremont, CA, USA: IETF, 2002.

[9] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The Eigentrust algorithm for reputation management in P2P networks," in *Proc. 12th Int. Conf. World Wide Web*, New York, USA, 2003, pp. 640–651.

[10] K. Hopkinson, X. Wang, R. Giovanini, J. Thorp, K. Birman, and D. Coury, "EPOCHS: A platform for agent-based electric power and communication simulation built from commercial off-the-shelf components," *IEEE Trans. Power Syst.*, vol. 21, no. 2, pp. 548–558, May 2006.

[11] X. R. Wang, K. M. Hopkinson, J. S. Thorp, R. Giovanini, K. Birman, and D. Coury, "Developing an Angent-based backup protection system for transmission networks," presented at the 1st Int. Conf. Power Syst. Commun. Syst, Infrastructures for the Future, Beijing, China, 2002.

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

[13] J. E. Fadul, K. M. Hopkinson, T. R. Andel, J. T. Moore, and S. H. Kurkowski, "Simple trust protocol for wired and wireless SCADA networks," in *Proc. 5th Int. Conf. Inf. Warfare Security*, Dayton, OH, USA, 2010, pp. 89–97.

[14] X. R. Wang, K. M. Hopkinson, J. S. Thorp, R. Giovanini, K. P. Birman, and D. V. Coury, "Developing an agent-based backup protection system for transmission networks," presented at the Power Syst. Commun. Syst. Infrastruct. Future Conf., Beijing, China, 2002.

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

[16] J. F. Borowski, K. M. Hopkinson, J. W. Humphries, and B. J. Borghetti, "Reputation-based trust for a cooperative agent-based backup protection scheme," *IEEE Trans. Smart Grid*, vol. 2, no. 2, pp. 287–301, Jun. 2011.

[17] K. P. Birman, *Reliable Distributed Systems: Technologies, Web Services, Applications*. New York: Springer, 2005.

[18] D. A. Rennels, "Fault-tolerant computing: Concepts and examples," *IEEE Trans. Comput.*, vol. C-33, no. 12, pp. 1116–1129, Dec. 1984.

[19] B. Parhami, "Voting algorithms," *IEEE Trans. Rel.*, vol. 43, no. 4, pp. 617–629, Dec. 1994.

[20] J. E. Fadul, K. M. Hopkinson, T. R. Andel, and J. T. Moore, "Trust management and security in the future communication-based "Smart" electric power grid," in *Proc. 44th Hawaii Int. Conf. Syst. Sci. (Elect. Power Syst.: Rel., Security, Trust Track)*, Koloa, Kauai, Hawaii, Jan. 4–7, 2011, pp. 1–10.

[21] J. E. Fadul, K. M. Hopkinson, T. R. Andel, J. T. Moore, and S. H. Kurkowski, "SCADA trust management system," in *Proc. Int. Conf. Security Manage. (Under The 2010 World Congr. Comput. Science, Comput. Eng., Appl. Comput.)*, Las Vegas, NV, USA, 2010, pp. 548–554.

[22] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Math.*, vol. 1, pp. 269–271, 1959.

[23] G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Sunnyvale, CA, USA: Insecure.Com LLC, 2008.

[24] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu, "Advances in network simulation," *IEEE Computer*, vol. 33, no. 5, pp. 59–67, May 2000.

[25] *"PSCAD/EMTDC Manual Getting Started,"* Manitoba HVDC Research Centre, Winnipeg, MB, Canada, 1998.

[26] *"PSS/E 30 User's Manual,"* Siemens Energy, Schenectady, NY, USA, 2004.

[27] *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Development Core Team, 2009.

[28] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, Modeling*. New York, USA: Wiley, 1991.

[29] V. Vittal, D. Martin, R. Chu, J. Fish, J. C. Giri, C. K. Tang, F. E. Vilaseca, and R. G. Farmer, "Transient stability test systems for direct stability methods," *IEEE Trans. Power Syst.*, vol. 7, no. 1, pp. 37–43, Feb. 1992.

[30] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3–4, pp. 591–611, Dec. 1, 1965.

[31] W. Mendenhall and T. Sincich, *Statistics for Engineering and the Sciences*, 4th ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1995.

[32] F. L. Ramsey and D. W. Schafer, *The Statistical Sleuth: A Course in Methods of Data Analysis*, 2nd ed. Pacific Grove, CA, USA: Duxbury, 2002.

Jose E. Fadul is a Course Director for the Software Professional Development Program (SPDP) at the Air Force Institute of Technology, Wright Patterson Air Force Base, OH, USA. His interests include software engineering and formal methods.

Kenneth M. Hopkinson (M'04–SM'10) is an Associate Professor of Computer Science at the Air Force Institute of Technology, Wright Patterson Air Force Base, OH, USA. His interests include simulation, networking, smart-grid protection systems, and security.

Todd R. Andel (M'07–SM'13) is an Associate Professor at the University of South Alabama, Mobile, AL USA. His is interested in secure embedded systems, networks, and formal methods.

Christopher A. Sheffield received the B.S. degree in computer and electronic engineering technology from the University of Dayton, Dayton, OH, USA, in 2010.

He is a staff member with the Air Force Institute of Technology's Center for Space Research and Assurance, Wright Patterson Air Force Base, OH. His interests include computer networks and security.