A Distributed Hierarchy Based Framework for Validating Edge Devices Performing State Estimation in a Power System

Chetan Kumar Kuraganti, Bryan Paul Robert, Gurunath Gurrala, Arun Babu Puthuparambil, Rajesh Sundaresan
Indian Institute of Science, Bangalore, India

***Abstract*—Recent cyber-attacks on power grids highlight the necessity to protect the critical functionalities vital for the safe operation of a grid. One such example is the power grid state estimation (SE), since various attacks can be launched by manipulating the SE results. In this paper, we propose a distributed hierarchy based framework to secure SE on edge devices. The data for SE is acquired from the phasor measurement units (PMUs) installed at various locations within the grid. These PMUs may be reprogrammed by a malicious actor to manipulate the data which may cause SE results to be inaccurate. Moreover, SE is carried out at a fixed central location, which makes it a prime target for cyber-attacks. Our proposed framework ensures that data aggregation and SE are carried out at a random device, and incorporates security features such as attestation and trust management to detect malicious devices. We test our proposed framework on a physical cluster of Parallella boards, monitoring a virtual IEEE 5 bus system. We also do simulations on the IEEE 118 bus system. Our simulations show that the trust for malicious devices nominally reduces with the number of attestations.**

***Index Terms*—Leader Election, Attestation, Trust Management, State Estimation, Kalman Filtering.**

I. Introduction

Power grids are potential targets for various kinds of attacks, given the massive economic and social disruptions that a widespread and prolonged loss of electricity could cause. An attacker may gain operational access to the control system and could disrupt the power grid's operation. For a summary of some recent cyber-security breaches, see [1]. The primary goals of the above attacks were to obtain operational access to the central coordinator controlling the respective power grids. This coordinator is usually fixed in most of the installations. Even though the processing is distributed and loss of one sub-area data can be handled by the coordinator, loss of the coordinator itself will lead to the loss of visibility and control of the entire system.

In recent years, edge computing in smart grid has received some attention [2]. The edge devices, capable of a wide range of functions, are often called intelligent electronic devices (IEDs). They are installed at substations, and are equipped with processors to handle data aggregation and other functionalities associated with measurement and protection.

If the role of coordinator is randomly switched among the existing IEDs, and the automatic detection of malicious agents, their isolation, and subsequent recovery is enabled, then the possibility of complete loss of system visibility and controllability can be avoided. With this motivation, we propose a distributed hierarchy based framework to achieve not only a randomization of the coordinator, chosen to perform the critical computations, but also integrity assessment of the agents in the system. The coordinator performs critical computations and operations that ensure large system visibility and control, through secure data aggregation from the edge devices (IEDs). Our focus is on the integration of various schemes to protect the critical functionality and a physical demonstration of its working.

A. Our Contributions

Our contributions can be summarized as follows.

- We propose a distributed hierarchy based framework to protect the state estimation process on the edge devices. The hierarchy is obtained through the leader election process described in section II-B. Leader election, attestation, and trust management work together to protect SE at the edge devices.
- We develop a simple leader election (LE) protocol to randomize the location of the coordinator among the devices. We assume that the network is completely connected, and the devices exchange information through broadcasting.
- We propose a distributed consensus-based trust management scheme, assuming that the number of malicious agents in the network is *strictly less than* $\frac{N}{2} - 1$, with N being the total number of agents in the network. Using the attestation-cum-consensus scheme, devices first identify malicious agents and then eliminate their data from the calculations. If the leader itself is malicious, a new leader will be elected automatically.
- We validate our proposed framework on a testbed cluster of IEDs (Parallella) that monitor a virtual IEEE 5 bus system.
- Simulations on the IEEE 118 bus system are used to verify the scalablity of our proposed framework.

B. Related Works and Connections to This Work

PMUs are time synchronized through GPS, and are capable of producing accurate phasor representations of voltage and current signals [3]. We use a Kalman filter for SE on PMU measurements which run in a centralized manner, but in one randomly chosen control center/IED. Our proposal involves the use of leader election implemented in a distributed way. An agent, among the many in the grid, is chosen to coordinate and perform SE. Our scheme, inspired by [4], not only tries to prevent malicious agents from hijacking the election process, but also ensures that each agent has an equal probability of

978-1-7281-6350-5/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on November 25,2020 at 02:43:09 UTC from IEEE Xplore. Restrictions apply.

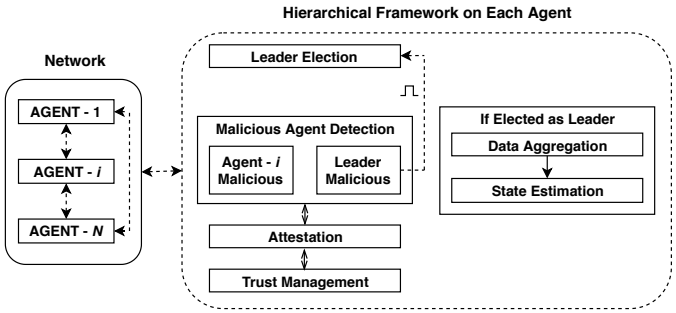


Fig. 1: Overview of the scheme.

being elected. Our choice of leader election scheme makes significant usage of commitment schemes; for details see [5].

In this work, we have used a distributed software-based attestation scheme, inspired by the one proposed in [6], to verify the integrity of the code running within an agent, without physically accessing it. Typically, software-based remote attestation uses a challenge-response based protocol between two agents, as described in SWATT [7]. If the integrity of the code running within an agent is compromised then we call it as malicious. In order to infer that there are malicious agents and to identify them, we utilize a trust management scheme. As suggested in [8], we integrate attestation with trust management, where attestation gives a measure of integrity, and trust management uses the outcome to determine whether an agent is malicious or not.

II. Methodology

In this section, we give an overview of our framework, depicted in fig. 1.

A. State Estimation

The SE algorithm estimates the state of the power system by processing the measurements obtained from various locations within the power system. In this work, we consider PMU measurements which provide voltage and current phasors, the voltage measurement model is linear, although the current phasor measurement model can be nonlinear if one represent system state in polar coordinates. By using the methods in [9] and [10], if one uses rectangular coordinates, the current phasor measurement model, too, is not only linear but also time invariant. Hence one gets the linear model

$$z_t = Hx_t + \eta_t, \tag{1}$$

where z_t and x_t are the measurement and the state vectors in rectangular coordinates, η_t is the noise vector written in rectangular coordinates, and H does not depend on time [10]. Furthermore, the state evolution can be written in rectangular coordinates as

$$x_{t+1} = x_t + \xi_t. \tag{2}$$

In rectangular coordinates, the state evolution noise vector ξ_t is modelled as $N(0, Q_t)$ and observation noise vector η_t is modelled as $N(0, R_t)$ [11]. In this work, we used Kalman filter based state estimation approach, described in [12]. In the following sections, we will illustrate the security schemes used in this work.

B. Leader Election

In our design, SE will be run at a leader. We assume that there may be multiple malicious agents, even during leader election, and the malicious agents may try to influence the outcome of election. For example, one of them may desire to be the leader, or avoid becoming the leader, or favor another malicious agent to become the leader. Furthermore, agents can communicate with each other only via broadcast, and no unicast or multicast is allowed during the election process. We assume that a secure broadcast channel is available and the broadcast messages are taken to be common knowledge. A detailed description of our scheme, shown in fig. 2, is provided below. Suppose there are N agents in the network.

- Each agent i chooses a 32-bit identity number, ID_i, which is broadcast to all agents. Once an agent receives $ID := [ID_0, ID_1, \ldots, ID_{N-1}]$, it will then sort them in ascending order, and store them in ID_{sorted}[1].
- In the next step, each agent chooses a number $C_i \in \{0, ..., N-1\}$ and a random string R_i.
- Agent i then commits C_i using a cryptographic hash function $\mathcal{H}$, which takes the hash of C_i appended with the random string to give $HC_i := \mathcal{H}(C_i||R_i)$.
- Agent i broadcasts its own hash HC_i, and aggregates the hashes $HC := [HC_0, \ldots, HC_{N-1}]$ of all agents.
- In the next phase, agents reveal their C_i and R_i to every other agent (via broadcast). This information, along with respective hashes, is used to verify that the agent has committed to its C_i, and has not changed it.
- Now, each agent will compute k, using the equation

$$k = \left(\sum_{i=0}^{N-1} C_i \right) \bmod N. \tag{3}$$

- Finally, the agent with the k^{th} smallest ID in ID_{sorted} is chosen as the leader.

Since each agent will possess ID_i from every agent i, ID_{sorted} will be identical across agents. Similarly, all agents will be able to arrive at the same value of k, since each agent will possess C_i from every agent i. They can therefore agree on a particular agent becoming the leader without the need for a central entity. These ID's and C's are unique whenever a new elections process is initialized. The above algorithm is a modification of the algorithm $A - LEAD^{ps,uni}$ in [4] which is used to elect a leader in an asynchronous unidirectional ring network of agents. Our version of algorithm differs from $A - LEAD^{ps,uni}$ in two ways. First, we use a hash-based commitment scheme instead of Naor's protocol [4]; second, $A - LEAD^{ps,uni}$ algorithm embeds a unidirectional ring into a completely connected network. However, this approach is time consuming, since any communication must pass through

[1]The USA has about 1700 PMUs installed. The probability that two agents have the same ID is upper bounded by 4×10^{-7}.

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on November 25,2020 at 02:43:09 UTC from IEEE Xplore. Restrictions apply.

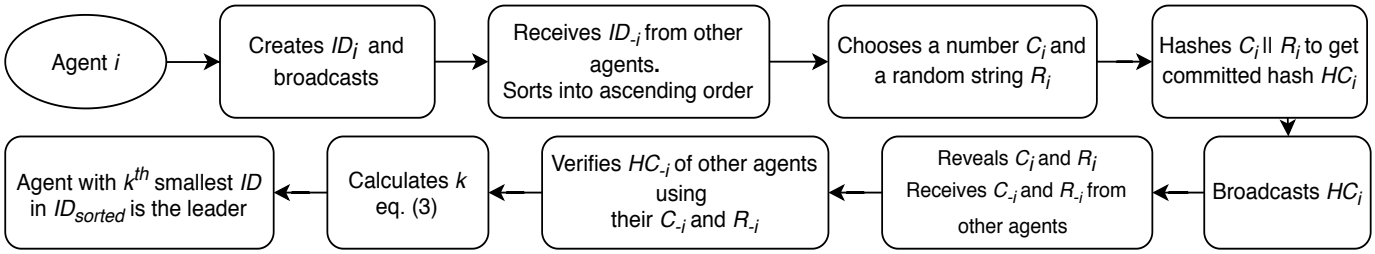


Fig. 2: Overview of the leader election scheme.

the entire ring. Instead, for ease of implementation, we allow agents to use a secure broadcast channel for communication, which makes our problem easier than the one in [4].

We choose a hash-based commitment scheme built on cryptographic hash functions . These commitment schemes are known to be secure; see [5]. The hash function we use in our work is taken from the libhydrogen cryptographic library [13].

C. Threat Model

In our model, we assume a cyber-attack in which an attacker can capture an agent or group of agents, re-program them with malicious code, and then re-deploy them back into network with the intent of affecting state estimation.

In our threat model we assume that the number of malicious agents is $< \frac{N}{2} - 1$ and the attacker does not enlarge the agents' memory. This makes it difficult to compromise the attestation algorithm itself. Further, we assume that the processor speed and memory access rate cannot be increased and a compromised agent stays compromised. Note that we do not consider physical attacks, such as transmission blockage, modification of physical sensors, etc., which are other ways by which an agent's integrity could be compromised.

D. Malicious Agent Detection

In this section, we first describe the attestation scheme. We then describe the trust management scheme. Together they help detect malicious agents in the network.

1) Attestation: Our attestation framework differs from others in using the concept of a *report*, i.e., after every attestation, the verifier broadcasts the details of attestation to all other agents. This report contains various parameters used in attestation, along with the outcome, i.e., whether the verifier suspects the attester to be malicious or otherwise. Agents make use of the reports to arrive at a consensus on whether malicious agents are present, and if yes, identify these malicious agents.

The attestation algorithm is designed to check if the program code has been corrupted by an attacker, as described in section II-C. The program memory, which contains the vital code is checked during the attestation. Every agent serves as a verifier at least once in a window of approximately T seconds. During this interval, an agent will provision itself as a verifier at a random instant of time, and challenge a randomly chosen attester, and then stay idle for the remainder of the time interval. However, in this interval, an agent can receive multiple challenges. This process is repeated every T seconds, ad infinitum. The frequency of attestations an agent can perform (the value of T) can be limited based on its computational capability.

The attestation scheme, shown in fig. 3, is summarised below.

- Once an attester is chosen, the verifier creates the challenge, and sends it to the attester. The challenge includes the following parameters:
  - A *nonce* that ensures that the challenge is not duplicated.
  - The *IP address* of the *attester*.
  - The random *offset*, which specifies the starting address of the memory region to be validated.
  - The *size* of the number of bytes to be validated.
  - The *timestamp* of when the challenge is issued.
  - The *signature* to ensure authenticity of the agent.
- The attester verifies the contents of the challenge, performs a *checksum* on the memory region specified by the challenge, and then provides a *response* containing the *hash* generated by the *checksum*.
- The verifier validates the response by calculating the hash on its own memory, using the same challenge parameters.
- If the hashes are identical, then the verifier finds the attester to be normal. Otherwise, it suspects the attester of being malicious. The verifier then broadcasts the *status* to the entire network.
- The other agents then update the trust of the attester based on the *status* of the report. See section II-D2 below.

2) Trust Management: Suppose there are N agents in the network. Whenever an agent k attests an agent j and broadcasts its report, the evolution of trust of agent j at agent i at time $t+1$ can be expressed as

$$p_{ij}(t+1) = \Big[p_{ij}(t) + \Delta_{k\to j}(p_{ik}(t))\Big]_0^1, \quad \forall i = 1, \ldots, N \quad (4)$$

where $[x]_0^1$ is the projection of x on the set $[0,1]$, $p_{ik}(t)$ is the trust of agent k at agent i at time t, and $\Delta_{k\to j}(p_{ik}(t))$ is (in our design)

$$\Delta_{k\to j}(p_{ik}(t)) = \begin{cases} \frac{p_{ik}(t)}{N}, & \text{if the attestation is positive,} \\ -\frac{p_{ik}(t)}{N}, & \text{if the attestation is negative,} \end{cases} \quad (5)$$

where the subscript $k \to j$ indicates that agent k is the verifier and agent j is the attester. When the agent j fails an attestation, we call it as a negative attestation, whereas if an attestation is

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on November 25,2020 at 02:43:09 UTC from IEEE Xplore. Restrictions apply.

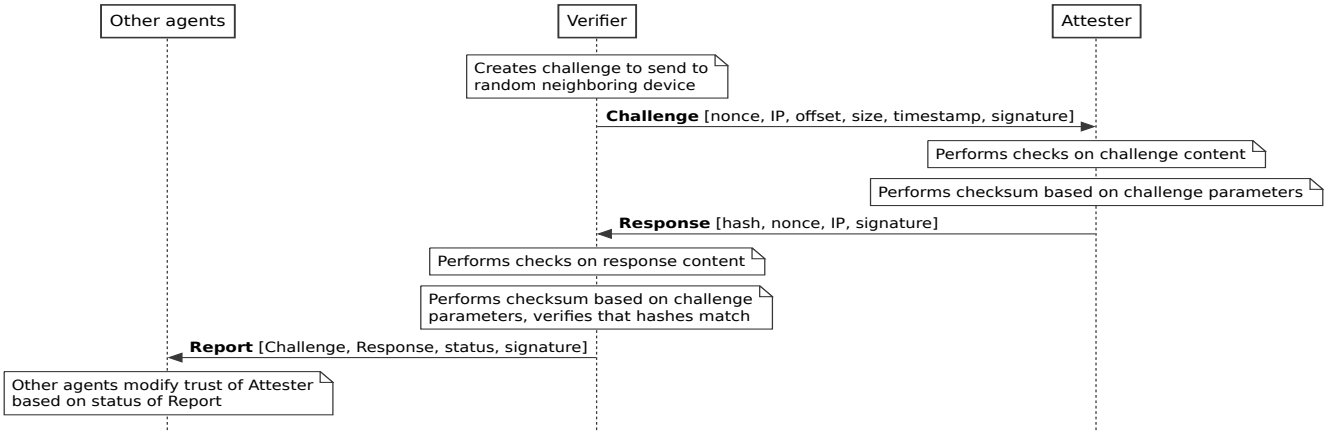


Fig. 3: Sequence diagram of the attestation scheme.

successful, we call it as a positive attestation. We assume that every agent has its own opinion of trust for every other agent in the network and must perform an attestation on another agent in a set time interval. Moreover, we also assume that there will be no report losses during the attestation process. At the beginning, an agent's opinion of trust for every agent in the network is initialized to 1.

Whenever an agent k attests agent j, agent k broadcasts the status to all other agents. Then, every agent i, $i \in \{1, \ldots, N\} \setminus \{j\}$, in the network, updates the trust of agent j using eq. (4) by considering its own opinion of trust of agent j and agent k, which are $p_{ij}(t)$ and $p_{ik}(t)$, respectively. If the opinion of trust of a particular agent reduces to 0, then agents perform majority voting to cast out the malicious agent from the network. Trust of a non-malicious agent may reduce because of false accusations by the malicious agents. However, other non-malicious agents increase the trust for that agent with every positive attestation. We use the factor $\frac{1}{N}$ to ensure graceful updates.

The algorithm above is a projected stochastic gradient descent algorithm and can be analyzed. For example, one can show that if we start with an initial condition when all trust values are 1 and the number of malicious agents is $< N/2-1$, then, with high probability, the trust values will settle at the point where all honest agents have a trust value of 1 and all others have a trust value of 0. The iterates of the trust update algorithm track a differential inclusion which can be analyzed. Due to space limitation, we do not include a formal statement in this conference version. Instead, we resort to simulations for validation.

III. Results

The algorithms are prototyped and tested on a development board called Parallella [14]. In our work, each IED is a Parallella which would record voltage and current data, gather data from other IEDs, and perform SE if elected as leader. In order to test our framework, a cluster of IEDs are used, as shown in fig. 4. The framework for communication between agents is created using serf [15], which allows broadcast and unicast communication. It provides a platform for devices to execute the challenge-response protocol, and gives them the capability to broadcast events and trigger responses.

For demonstration purposes, due to cost considerations, 5 Parallella boards were used to represent IEDs of the IEEE 5 bus system, as shown in fig. 5, and are connected in a star network. Initially, devices are started-up almost simultaneously, so that they can be synchronised. Then, they start broadcasting information to each other, so that each device can have a view of the network, which is designed using serf. Once they become part of the serf cluster, a leader will be elected, as explained in section II-B. Now, devices start sending data to the leader periodically (once a minute).

Additionally, simulations are performed on the IEEE 118 bus system to test the scalability of our proposed framework.

A. SE when there is no malicious agent

We perform SE on IEEE 5 bus system. Virtually, we map each available Parallella to each bus in the 5 bus system shown in fig. 5. The data for SE—line and bus parameters—is acquired from MATPOWER toolbox. In our framework, SE will begin right after the leader election scheme. The estimated phase angles are shown in fig. 6f without any malicious agent. In fig. 6f, the red line indicates true state of the system, whereas the blue line indicates the estimated states. This result is obtained after a single Kalman iteration, and the squared error (L_2-norm of the error vector) observed is 0.147.

Fig. 4: IED cluster.

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on November 25,2020 at 02:43:09 UTC from IEEE Xplore. Restrictions apply.

B. SE when agent at Bus-3 is malicious

We deliberately make the agent at bus-3 malicious by modifying its core process. After the leader election, attestation between devices begins and the trust of a device is updated based on the reports obtained from attestation. In our implementation, every device randomly performs an attestation in a 40 second interval. Whenever trust of a device drops to 0 at majority of devices, data from that device is ignored. Our combined choice of attestation and trust management is able to detect the malicious device in near real time. The fig. 6 shows the evolution of trust at each agent about all the other agents, using eq. (4). From fig. 6 one can observe that opinion of trust for agent 3 is reduced to 0 at agents 1, 2, 4, and 5, presented in figs. 6a, 6b, 6d and 6e, respectively. However its own opinion of trust, in fig. 6c is very high—because it doesn't want to become a target by reducing its own trust. Now, through majority voting, agent 3 will be identified as malicious and its measurements will be subsequently ignored by the leader. Also, it is evident from figs. 6a to 6e that trust for the other agents is varying, since agent 3 is constantly accusing others of being malicious through negative attestations. However, their trust increases and reaches to 1 due to positive attestations from non-malicious agents. This process is constantly running in the background to track any suspicious activity.

The SE result of this case is presented in fig. 6g. The squared error observed in this case—after a single Kalman iteration—is 0.2034. During the simulation, data from device 3 is ignored. In both the cases, we were able to obtain accurate estimates. This process is can be scaled to bigger systems.

In fig. 6h, we provide maximum absolute error observed in the system when a malicious agent is present. The simulation is carried out for a duration of 40 samples. A malicious agent is introduced at the 21^{st} sample, and is detected at the 28^{th} sample; note that the observed error in the system is elevated because of the fabricated data reported by the malicious agent. Once detected, we ignore the data from malicious agent.

C. Multiple malicious agents

1) IEEE 5 Bus System: Evolution of trust for multiple malicious agents in the IEEE 5 bus system, fig. 5, at a non-malicious agent, is presented in fig. 7a and fig. 7b. The evolution of trust at a non-malicious agent when malicious agents do not cooperate with each other is presented in fig. 7a. However, these malicious agents may cooperate each other to increase their own trust. The results for this case is given in fig. 7b. In this case, we assume that a malicious agent always chooses a malicious agent as an attester whenever it gets a chance of being a verifier and always reports in favor of a malicious agent. In addition to this, when a non-malicious agent reports against one of the malicious agent, all malicious agents increase the trust of this agent instead of reducing it. In both cases, our framework is able to identify all the malicious agents in the system, even though the number of malicious is $\not< N/2 - 1$. Furthermore, from fig. 7a and fig. 7b, one can observe that detection of malicious agents when they are cooperating with each other takes more number of attestations in comparison to non-cooperating case.

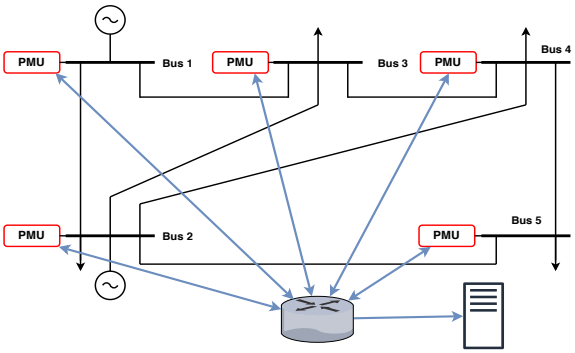


Fig. 5: IEEE 5 bus system with PMUs, a star network.

2) IEEE 118 Bus System: The IEEE 118 bus system is used to test the scalability of our proposed framework and the group of 5 IEDs at buses 45, 46, 47, 48 and 49 are assumed to be malicious. Considering the cost constrains involving with Parallella devices, we resort to only simulations on this test system to evaluate our framework. Simulation results on 118 bus systems for both non-cooperative and cooperative cases are presented in fig. 7c and fig. 7d, respectively. Though our framework is able to identify all the malicious agents, the number of attestations performed in both the cases are high. For a bigger system with hundreds of buses, our framework might take some time to reach a consensus through majority voting. Therefore, suitable methods to reduce the number of attestations are being explored as on of the future directions.

IV. Conclusion

In this paper we propose a distributed hierarchy based framework to protect a critical function (SE) of a coordinator in a distributed environment. The protection comes from choosing the coordinator agent in a random fashion, from an attestation and trust management protocol that maintains and updates the trust metric of every agent in the system, and from a consensus algorithm that isolates the malicious agents. Our main focus has been to demonstrate a decentralized framework to protect a critical function (SE).

Acknowledgement

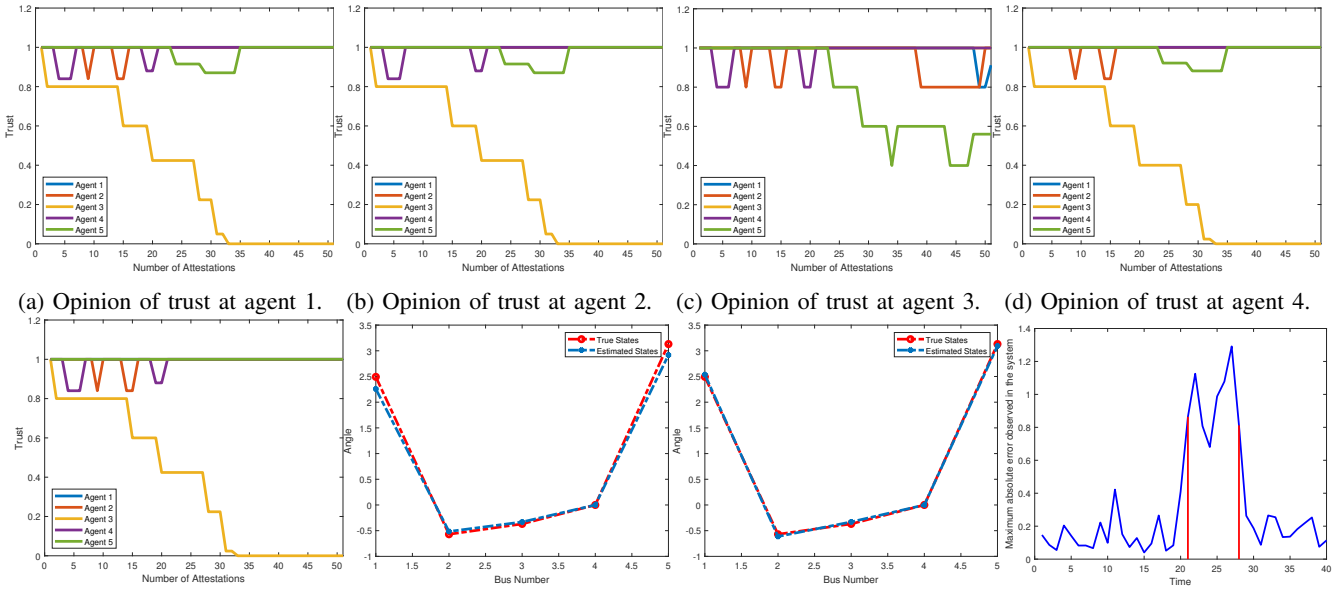
The authors would like to thank Dr. Himanshu Tyagi for valuable discussions. This work was supported in part by the Bosch Research and Technology Centre, Bengaluru, India and by the Robert Bosch Centre for Cyber-Physical Systems, Indian Institute of Science, Bengaluru, India, (under Project E-Sense: Sensing and Analytics for Energy Aware Smart Campus), and in part by the Science and Engineering Research Board, Department of Science and Technology (grant no. EMR/2016/002503).

References

[1] N. Kshetri and J. M. Voas, "Hacking power grids: A current problem," *Computer*, vol. 50, pp. 91–95, 2017.

[2] F. Samie, L. Bauer, and J. Henkel, *Edge Computing for Smart Grid: An Overview on Architectures and Solutions*. Cham: Springer International Publishing, 2019, pp. 21–42. [Online]. Available: https://doi.org/10.1007/978-3-030-03640-9-2

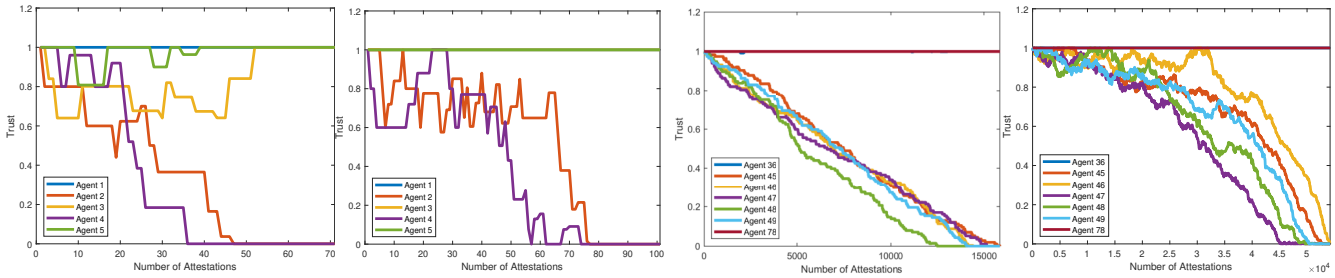
Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on November 25,2020 at 02:43:09 UTC from IEEE Xplore. Restrictions apply.



(a) Opinion of trust at agent 1. (b) Opinion of trust at agent 2. (c) Opinion of trust at agent 3. (d) Opinion of trust at agent 4.

(e) Opinion of trust at agent 5. (f) State Estimation without malicious agent. (g) State Estimation with one malicious agent. (h) Maximum absolute error observed in the system.

Fig. 6: Trust at individual agents and results of state estimation with and without malicious agents.



(a) Opinion of trust at non-malicious agent in IEEE 5 bus system with two non-cooperative malicious agents.

(b) Opinion of trust at non-malicious agent in IEEE 5 bus system with two cooperative malicious agents.

(c) Evolution of trust at one non-malicious agent in the IEEE 118 bus system on five non-cooperative malicious agents.

(d) Evolution of trust at non-malicious agent in IEEE 118 bus system on five cooperative malicious agents.

Fig. 7: Evolution of trust at a non-malicious agent with multiple malicious agents.

[3] A. Gomez-Exposito, A. Abur, P. Rousseaux, A. de la Villa Jaen, and C. Gomez-Quiles, "On the use of pmus in power system state estimation," in *17th Power Systems Computation Conference 2011*. Stockholm, Sweden: 17th Power System Computation Conference, 2011, pp. 1080–1092.

[4] I. Abraham, D. Dolev, and J. Y. Halpern, "Distributed protocols for leader election: A game-theoretic perspective," in *Distributed Computing*, Y. Afek, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 61–75.

[5] I. Damgard and J. B. Nielsen, "Commitment schemes and zero-knowledge protocols, 2006," 2008.

[6] Y. Yang, X. Wang, S. Zhu, and G. Cao, "Distributed software-based attestation for node compromise detection in sensor networks," in *Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems*, ser. SRDS '07. USA: IEEE Computer Society, 2007, p. 219–230.

[7] A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla, "Swatt: software-based attestation for embedded devices," in *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*. Berkeley, CA, USA: IEEE, May 2004, pp. 272–282.

[8] J. Lopez, R. Roman, I. Agudo, and C. Fernandez-Gago, "Trust management systems for wireless sensor networks: Best practices," *Computer Communications*, vol. 33, no. 9, pp. 1086–1093, 2010.

[9] J. Chen, "Measurement enhancement for state estimation," Ph.D. dissertation, Texas A&M University, 5 2008.

[10] R. Nuqui, "State estimation and voltage security monitoring using synchronized phasor measurements," Ph.D. dissertation, Virginia Polytechnic Institute, 07 2001.

[11] A. S. Debs and R. E. Larson, "A dynamic estimator for tracking the state of a power system," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-89, no. 7, pp. 1670–1678, Sep. 1970.

[12] A. Monticelli, *State Estimation in Electric Power Systems: A Generalized Approach*. New York, US: Springer US, 01 1999.

[13] F. Denis, "A lightweight, secure, easy-to-use crypto library suitable for constrained environments." https://github.com/jedisct1/libhydrogen, 2019.

[14] A. Olofsson, T. Nordström, and Z. Ul-Abdin, "Kickstarting high-performance energy-efficient manycore architectures with epiphany," in *2014 48th Asilomar Conference on Signals, Systems and Computers*. Pacific Grove, CA, USA: IEEE, Nov 2014, pp. 1719–1726.

[15] HashiCorp, "Serf," HashiCorp, 2019. [Online]. Available: http://serf.io/

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on November 25,2020 at 02:43:09 UTC from IEEE Xplore. Restrictions apply.