Analyzing Malicious Data Injection Attacks on Distributed Optimal Power Flow Algorithms

Mohannad Alkhraijah,* Rachel Harris,* Samuel Litchfield,† David Huggins,† and Daniel K. Molzahn*
* School of Electrical and Computer Engineering, Georgia Institute of Technology
† Georgia Tech Research Institute, Georgia Institute of Technology
Atlanta, GA, USA

***Abstract*—The rapid growth of distributed energy resources motivates the application of *distributed optimization* algorithms for solving optimal power flow (OPF) problems. Using distributed optimization to solve OPF problems requires repeated data sharing between agents responsible for different regions of the power system. The performance of distributed optimization algorithms depends on the integrity of the shared data and the communications between the agents. This paper investigates the vulnerability of a distributed algorithm called Auxiliary Problem Principle (APP) with respect to cyberattacks on the communications between the agents. We propose cyberattack models that manipulate the shared data between neighboring agents to achieve an objective such as driving the algorithm to converge to a suboptimal solution. We investigate the convergence characteristics of the APP algorithm in the context of the DC optimal power flow (DC OPF) problem with and without manipulating the shared data. Last, we demonstrate how trained neural networks can provide highly accurate attack detection.**

***Index Terms*—Cybersecurity, Distributed Optimal Power Flow.**

I. Introduction

With rapid growth in distributed energy resources (DERs) and expanding communication infrastructures, the risk of cyberattacks challenges the security of power systems [1–3]. Much of the current cybersecurity research focuses on the existing centralized operational paradigm, but power systems are transitioning towards a more decentralized paradigm [4]. Distributed algorithms help address the computational complexity and data sharing challenges associated with coordinating DERs. Finding the optimal operating point of DERs requires solving optimal power flow (OPF) problem. In distributed OPF algorithms, agents compute optimal setpoints for their region of the power system by sharing the computational task with their neighboring agents. Although distributed algorithms can potentially increase the resiliency of power system operations by eliminating a central single point of failure, involving many entities in controlling power system operations increases the number of potential targets for cyberattacks.

The vulnerability of power systems to data manipulation such as false data injection has been extensively discussed in the literature [3]. False data injection can severely impact distributed algorithm convergence [5, 6]. The impact and detection of false data injection attacks on distributed optimization algorithms has been considered in the context of generation control [7, 8], state estimation [9, 10], economic dispatch [11], and distributed energy management systems [12]. However, there is limited research on cyberattacks that manipulate data shared between agents in distributed OPF algorithms. References [13–15] investigate false data injection attacks on primal-dual gradient descent distributed DC OPF algorithms. They also propose a detection method which uses information estimation on data from previous iterations to detect malicious shared data. In all these references, the authors consider a simplistic attack scenario where the attacker shares data directly corresponding to the attacker's target operating point. However, more sophisticated attack scenarios could bypass previously proposed detection methods. For example, an attack which begins with the first iteration or quickly converges to the attacker's target would prevent defenders from collecting enough historical data to perform the information estimation. We begin to explore a more sophisticated feedback control attack and present initial results for a machine learning detection algorithm in [16].

In this paper, we investigate the vulnerability of the Auxiliary Problem Principle (APP) algorithm to adversarial attacks on the shared data. We propose three cyberattack models for manipulating the shared data during the distributed optimization process: (1) sending the attacker's target values in every iteration, (2) using PID feedback control, and (3) using bilevel optimization. In the first two models, the adversarial agent identifies a target solution to the OPF problem and uses this information to send false values for the shared variables. In the third model, the adversarial agent solves a bilevel optimization problem with a target objective in the upper level and the neighboring region's problem in the lower level. With these attack strategies, we show that an attacker can drive the distributed optimization algorithm to a feasible target solution. Next, we explore how convergence patterns change for algorithms under attack, and use neural networks (NN) trained on shared data to detect attacks with high accuracy.

The rest of the paper is organized as follows: Section II describes the problem formulation and the distributed APP algorithm. Section III provides the proposed cyberattack models. Section III presents numerical results and discusses the attack implementation and detectability. Finally, Section V concludes the paper and describes future work.

II. Problem Formulation

The OPF problem is fundamental to power system operations. OPF problems search for optimal operating points which satisfy constraints from both the network model (the power flow equations) and engineering constraints (transmission lines' thermal limits, generator outputs' limits, etc). The OPF problem is a non-convex optimization problem due to the nonlinearity of the power flow equations. To guarantee the convergence of the APP algorithm, we use the DC OPF approximation of the power flow (DC OPF) [17] to investigate the vulnerabilities of distributed algorithms, since the convergence guarantee of the APP algorithm is limited to convex problems. In our future work, we plan to extend these results to include more sophisticated power flow representations.

This section first describes the centralized DC OPF formulation and then presents the APP algorithm which solves DC OPF problems in a distributed fashion.

A. DC Optimal Power Flow

Consider a system with sets of buses, lines, and generators denoted by $\mathcal{N}$, $\mathcal{L}$, and $\mathcal{G}$. The DC OPF formulation is:

$$\min \sum_{g\in\mathcal{G}} f_g(p_g) \tag{1a}$$

subject to:

$$p_i - d_i = \sum_{(i,j)\in\mathcal{L}} B_{ij}(\theta_i - \theta_j), \qquad \forall i \in \mathcal{N}, \tag{1b}$$

$$P_g^{min} \le p_g \le P_g^{max}, \qquad \forall g \in \mathcal{G}, \tag{1c}$$

$$-P_{ij}^{max} \le B_{ij}(\theta_i - \theta_j) \le P_{ij}^{max}, \qquad \forall (i,j) \in \mathcal{L}, \tag{1d}$$

where f_g is the cost function and p_g is the power output of generator $g \in \mathcal{G}$. B_{ij} denotes the admittance and P_{ij}^{max} denotes the thermal limit of the line $(i,j) \in \mathcal{L}$. We define the state of the buses with the phase angle of the voltage, θ_i for bus $i \in \mathcal{N}$. We denote the demand at bus $i \in \mathcal{N}$ as d_i. The optimization problem minimizes the total generation cost as shown in the objective function (1a), subject to the operation constraints (1b)–(1d). The constraint (1b) is the DC power flow equations, while constraints (1c) and (1d) are the generators' power output limits and the transmission lines' thermal limits.

B. Distributed DC Optimal Power Flow

We next describe the Auxiliary Problem Principle (APP) distributed algorithm that we use to solve the DC OPF problem. To solve the OPF problem using the APP algorithm, we decompose the power system into regions. Each region is equipped with an agent that is capable of solving an optimization problem and communicating with neighboring agents. Let $\mathcal{N}_m$, $\mathcal{L}_m$, and $\mathcal{G}_m$ define the sets of buses, lines, and generators for region $m \in \mathcal{M}$, where $\mathcal{M}$ is the set of all regions. Two neighboring regions are connected through tie-lines that belong to both regions. For each tie-line between two regions, we introduce dummy variables at each terminal of the tie-line that represent the state variables, i.e., the voltage angles θ in the DC OPF problem, and enforce the dummy variables to be equal to the original variables using consistency constraints. We denote the set of shared variables in region m with $\mathcal{N}_m^s$. The network decomposition is described in Figure 1.

By relaxing the consistency constraints using the Augmented Lagrangian method and applying the APP algorithm, the DC OPF problem is then iteratively solved using a

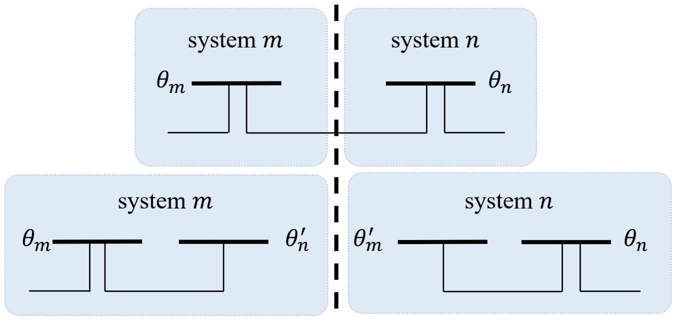

Fig. 1. Illustrative example of the network decomposition.

sequence of smaller problems [18]. Each problem consists of an agent's local objective function, local constraints, and relaxed consistency constraints with neighboring agents. The consistency constraints are evaluated using the values received from neighboring agents in the previous iteration, so each local problem can be solved independently. The local problem corresponding to region m at iteration $k+1$ is:

$$\begin{aligned}\min_{p^{k+1},\theta^{k+1}} \quad & F_m^{k+1}(\theta_m^k, \theta_n^k, \lambda^k) \\ &= \sum_{g\in\mathcal{G}_m} f_g(p_g^{k+1}) + \frac{\beta}{2}||\theta_m^{k+1} - \theta_m^k||_2^2 \\ &\quad + \gamma(\theta_m^{k+1})^\top(\theta_m^k - \theta_n^k) + (\lambda^k)^\top \theta_m^{k+1}\end{aligned} \tag{2a}$$

subject to:

$$p_i^{k+1} - d_i = \sum_{(i,j)\in\mathcal{L}_m} B_{ij}(\theta_i^{k+1} - \theta_j^{k+1}), \qquad \forall i \in \mathcal{N}_m, \tag{2b}$$

$$P_g^{min} \le p_g^{k+1} \le P_g^{max}, \qquad \forall g \in \mathcal{G}_m, \tag{2c}$$

$$-P_{ij}^{max} \le B_{ij}(\theta_i^{k+1} - \theta_j^{k+1}) \le P_{ij}^{max}, \quad \forall (i,j) \in \mathcal{L}_m, \tag{2d}$$

where α, β, and γ are user-selected parameters associated with the APP algorithm. F_m is the local objective function of agent m. The variables θ_m^k and θ_n^k are vectors, with the same cardinality as $\mathcal{N}_m^s$, that denote the shared variables' values obtained from the solution of the same agent's and the neighboring agents' local problems from the previous iteration, respectively. λ^k is a vector of Lagrange multipliers. The superscript $\top$ denotes the vector transpose, and $||\cdot||_2$ denotes the vector l_2-norm. In each iteration, the agents first solve their local problems (2) in parallel. Then, each agent sends the shared variables θ_m^{k+1} to the neighboring agents. Using the received values, the agents then update the Lagrange multipliers λ using (3):

$$\lambda^{k+1} = \lambda^k + \alpha(\theta_m^{k+1} - \theta_n^{k+1}), \tag{3}$$

where θ_n^{k+1} denotes the shared variable values received from the neighboring agents. Once the agents calculate the Lagrange multipliers, they again solve their local problems and share the results of the shared variables with their neighbors. The algorithm terminates when the l_2-norm of the mismatch between the shared variables is less than a specified tolerance.

III. Threat Model and Attacker Strategies

We model an attacker as an agent that controls a particular region in order to manipulate the APP algorithm. The attacker

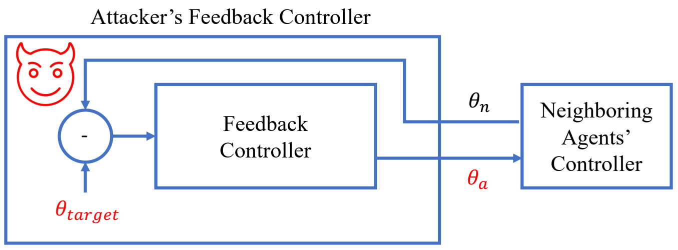
Attacker's Feedback Controller
-
Feedback Controller
θ_{target}
θ_n
θ_a
Neighboring Agents' Controller

Fig. 2. Feedback control attack model.

has complete knowledge of the entire power system, including both the system-wide problem and the problems solved by neighboring agents. We primarily consider a financially motivated attacker. However, our threat model can be readily extended for other purposes such as increasing the system's operating costs or causing constraint violations that damage physical components. In this paper, the attacker seeks to increase the revenue by forcing the APP algorithm to converge to a solution where other regions buy more power from the attacker's region. First, the attacker finds an advantageous convergent state for the system and then determines an approach for driving the shared variable values such that the APP algorithm converges to that state. We propose three methods to accomplish this with varying levels of complexity and detectability, e.g., how much the attack deviates from the typical behavior of the APP algorithm.

1) Simple Attack: The attacker determines an advantageous convergent state by modeling the system-wide OPF problem, but manipulates generation cost functions such that neighboring regions purchase more power from the attacker's region. This provides the attacker with the voltage angles at the tie-lines to neighboring regions that would correspond with this solution. With this knowledge, we first consider an attack strategy where the malicious agent sets the shared variables they control to exactly these target voltage angle values at each iteration of the algorithm. This is the least complex approach, but also the least subtle, increasing the likelihood of detection.

2) Feedback Controller: To manipulate the shared data in a less obvious manner, we next consider an attack strategy that incorporates a feedback control mechanism. The attacker uses a Proportional-Integral-Derivative (PID) controller to drive the shared variable values to their target values. This casts the attacker's problem as a traditional feedback control problem, with the shared variable values received from neighboring agents representing inputs, the neighboring regions acting as plants, and the shared variable values sent by the attacker functioning as actuators. This is illustrated in Figure 2.

The attacker determines their shared variable values at each iteration k by first calculating the error e_k between the neighboring shared variables, θ_n^k, and their target, θ_{target}^k as $e_k = \theta_{target} - \theta_n^k$. The attacker then computes an actuation term using three tuned parameters (a term proportional to the error K_p, a term related to the accumulated error K_i, and a term related to the rate of change in the error K_d) as shown in (4) to obtain the attacker's shared variable values for the next iteration, θ_a^{k+1}.

$$\theta_a^{k+1} = \theta_{target} + K_p e_k + K_i \sum_{n=0}^{k} e_n + K_d(e_k - e_{k-1}). \quad (4)$$

Varying the parameters K_p, K_i, and K_d changes the dynamics of the APP algorithm's convergence and can possibly cause the algorithm to not converge to the target at all.

3) Bilevel Optimization: The attacker solves a bilevel optimization problem with a target objective in the upper level and the neighboring agents' problems in the lower level. The advantage of this model is that the attacker does not need to specify target operating points in advance. Rather, the attacker's bilevel problem searches for the optimal solution that is obtainable given their ability to manipulate particular shared variable values as well as system-specific characteristics such as the network structure, electrical parameters, etc.

To successfully manipulate the shared variable values, the attacker needs to consider multiple consecutive iterations in the bilevel optimization. The first iteration corresponds to the original local problem described in (2) without any manipulation to identify the neighboring agents' outputs for the next iteration. In the subsequent iterations, the attacker tries to select shared variables that will force the neighboring agents' local solutions to match the attacker's goals as closely as possible. For a setting where the attacker considers two iterations, the bilevel optimization problem is:

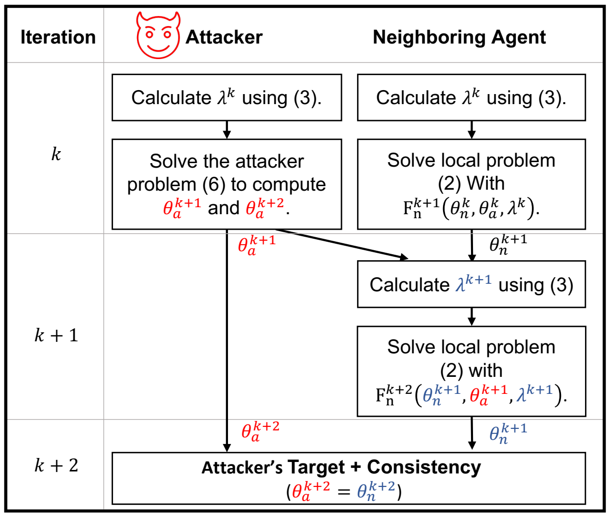
Iteration
Attacker
Neighboring Agent
k
$k+1$
$k+2$
Calculate λ^k using (3).
Calculate λ^k using (3).
Solve the attacker problem (6) to compute θ_a^{k+1} and θ_a^{k+2}.
Solve local problem (2) With $F_n^{k+1}(\theta_n^k, \theta_a^k, \lambda^k)$.
θ_a^{k+1}
θ_n^{k+1}
Calculate λ^{k+1} using (3)
Solve local problem (2) with $F_n^{k+2}(\theta_n^{k+1}, \theta_a^{k+1}, \lambda^{k+1})$.
θ_a^{k+2}
θ_n^{k+1}
Attacker's Target + Consistency
$(\theta_a^{k+2} = \theta_n^{k+2})$

Fig. 3. Flow diagram visualizing the bilevel attack strategy.

$$\min \quad G_{target}(p_a^{k+2}, \theta_a^{k+2}) \quad (5a)$$

subject to:

$$(p_n^{k+1}, \theta_n^{k+1}) = \underset{(p_n^{k+1}, \theta_n^{k+1}) \in A_n}{\arg\min} F_n^{k+1}(\theta_n^k, \theta_a^k, \lambda^k), \quad (5b)$$

$$(p_n^{k+2}, \theta_n^{k+2}) = \underset{(p_n^{k+2}, \theta_n^{k+2}) \in A_n}{\arg\min} F_n^{k+2}(\theta_n^{k+1}, \theta_a^{k+1}, \lambda^{k+1}), \quad (5c)$$

$$(p_a^{k+2}, \theta_a^{k+2}) \in A_a, \quad (5d)$$

$$\theta_a^{k+2} = \theta_n^{k+2}, \quad (5e)$$

$$\lambda^{k+1} = \lambda^k + \alpha(\theta_n^{k+1} - \theta_a^{k+1}), \quad (5f)$$

where G_{target} is the attacker's objective function. In this attack model, we consider a linear target objective function. We denote the attacker's and the neighboring agents' variables with subscript a and n, respectively. For notational simplicity, we denote the feasible region of each agent m as A_m defined by the constraints (2b)–(2d). Constraints (5b) and (5c) correspond to the lower-level problem consisting of two iterations of the neighboring agents' local problems. The representations of each iteration require the shared variables and the Lagrange multiplier values from the previous iteration as inputs to evaluate the objective functions F_n^{k+1} and F_n^{k+2}. Unlike (5b) where the inputs of F_n^{k+1} are constants, the inputs of the objective function F_n^{k+2} in (5c) contain decision variables. The objective functions of both lower-level problems, F_n^{k+1} and F_n^{k+2}, are quadratic functions while the constraints defined by the region A_n are linear.

We include the first iteration of the lower-level problem in (5) to simplify the representation of the attacker's problem. However, the attacker solves the first iteration of the lower-level problem (5b) before solving the bilevel problem. In other words, since θ_n^k, θ_a^k, and λ^k are inputs to (5b), we can evaluate (5b) prior to solving (5). We then solve the bilevel problem (5) by reformulating the second iteration's lower-level problem (5c) using its Karush–Kuhn–Tucker (KKT) conditions parameterized with the upper-level variables, i.e., θ_a^{k+1}. The parameterized KKT conditions consist of linear constraints with integer variables that we use to linearize the complementary slackness constraints [19]. Thus, the bilevel problem is a Mixed-Integer Linear Programming (MILP) problem that can be solved using commercial solvers.

Figure 3 illustrates the bilevel attack strategy. This figure depicts two iterations of the APP algorithm. The attack in the diagram starts at iteration k, where the attacker solves the bilevel optimization problem to compute the values that will be shared with neighboring agents, i.e., θ_a^{k+1} and θ_a^{k+2}, shown in red. These values drive the neighboring agents' solution to the targeted solution by changing the Lagrange multipliers and the second iteration outputs, i.e., λ^{k+1} and θ_n^{k+2}, shown in blue. This attack model ensures that the attacker's target setpoints are achieved and the consistency constraints are met after the second iteration.

IV. Attack Detection

We develop a distributed attack detection method, in which each local agent trains a NN to classify a sequence of shared variable mismatches as either "not attacked" or "attacked". The shared variable mismatch is the difference between the local agent's value for a shared variable and the neighboring agent's value. The sequences of shared variable mismatches exhibit different characteristics for different attack strategies. Therefore, each agent trains separate NNs for each attack strategy. The agent can then flag an attack if any of the NNs detect malicious data manipulation. After offline training, a NN can perform classification on shared variable mismatches very quickly during real-time operation.

Each agent records their shared variable mismatches from the final 50 iterations of the distributed algorithm to use as the input to the NN. The output is binary, with '1' representing attacked algorithms and '0' representing non-attacked algorithms. The NN contains 8 hidden layers, each with rectified linear unit (ReLU) activation. The loss function minimized during training is the mean absolute error, written as $\frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|$ for a set of N predictions, where $\hat{y}_i$ denotes the i-th prediction and y_i denotes the i-th true value. Before training, the data is normalized so that mismatches at each iteration have zero mean and unit variance.

V. Numerical Results

In this section, we show simulation results from implementing the proposed attack strategies on the APP algorithm.

A. Simulation Setup

We simulate attacks for three test systems: the IEEE 14-bus system, the IEEE 39-bus system, and the IEEE 118-bus system from [20]. We decompose the test cases into two regions controlled by two different agents. The attacker controls the output of one of the agents, while the other agents solve the distributed algorithm normally. The attacker's objective is to increase the total power output of the generators within the attacker's region in order to increase this region's revenues by selling more energy. As described by the threat models in Section III, the attacker can manipulate the distributed optimization algorithm only by modifying the values of the shared variables sent to the other agent.

We run the simulations using the Julia programming language [21] on a PC with an Intel Core i7 processor and 16 GB of RAM. We use JuMP [22] and Gurobi to solve all optimization problems and we use BilevelJuMP [23] to reformulate the bilevel optimization problems (5) as MILPs.

In the simple attack and PID feedback attack strategies, the attacker needs to find a target setpoint prior starting the attack. We determine these target setpoints by solving a centralized DC OPF problem using PowerModels [24] with a cost function that maximizes the total generation within the attacker's region. We tune the parameters of the APP algorithm to be $\alpha = \frac{\beta}{2} = \gamma = 2 \times 10^4$ and set the stopping tolerance to 10^{-4} radians.

B. Simulation Results

The three cyberattack models successfully drive the results of the APP algorithm to the attacker's target values. We use the *optimality gap*, i.e., the relative value of the objective function to the optimal solution, to measure the deviation of the attacker's target from the optimal solution. The results of the attack strategies using the three test cases are summarized in Table I. Figure 4 shows the convergence of the IEEE 118-bus test case to a suboptimal solution using the proposed attack strategies. The figure shows the operation cost of the true optimal solution and the relative cost of the suboptimal solution which is the attacker's target.

For both the simple attack and the PID attack, the APP algorithm converges after a few iterations to a value close to the target value as shown in Figure 4. However, for both of these attacks, it takes a larger number of iterations for the

TABLE I
SIMULATION RESULTS OF THE ATTACK MODELS

Case	Attack Model	Iterations	Attacker Output (MW)	Neighboring Region Output (MW)	Optimality Gap (%)	Computation Time (sec)
14-Bus	No Attack	80	0	259	0.02	0.99
	Simple Attack	1409	200	59	24.37	9.68
	PID Feedback Attack	981	200	59	24.39	6.65
	Bilevel Optimization Attack	51*	200	59	24.41	1.81
39-Bus	No Attack	397	3192	3061	0.06	5.66
	Simple Attack	648	3709	2544	8.12	5.26
	PID Feedback Attack	447	3709	2544	8.14	3.43
	Bilevel Optimization Attack	301*	3706	2548	8.08	5.86
118-Bus	No Attack	184	1043	3199	0.01	3.90
	Simple Attack	3735	2576	1666	20.25	53.56
	PID Feedback Attack	2778	2576	1666	20.25	38.56
	Bilevel Optimization Attack	101*	2576	1666	20.25	9.78

* The bilevel optimization attack converges within two iterations of starting the attack, the timing of which can be selected by the attacker.

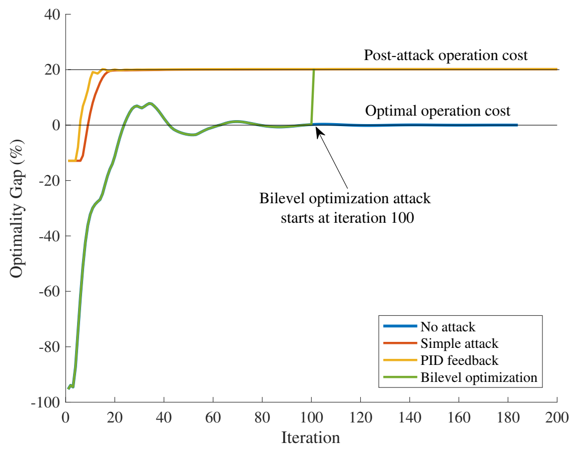

Fig. 4. 118-bus relative cost convergence pre- and post-attack.

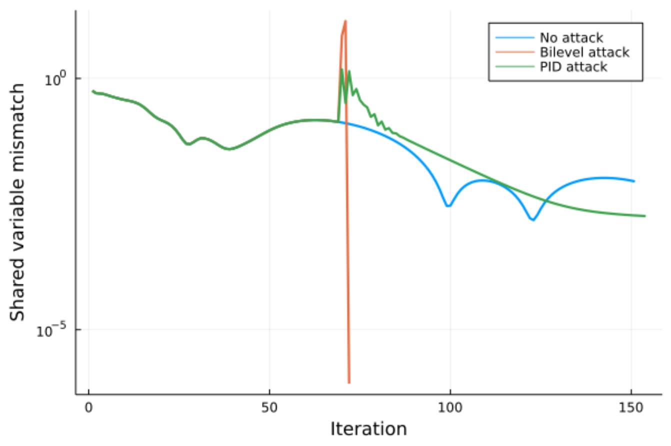

Fig. 5. Shared variable mismatches under no attack, bilevel attack, and PID attack.

shared variable mismatches to reach the stopping tolerance. Compared to the simple attack, we also observe that the PID feedback attack converges substantially faster in the three test cases as indicated in Table I. The PID attack takes 31%, 32%, and 26% fewer iterations to converge compared to the simple attack for the 14-bus, 39-bus and 118-bus test cases, respectively. The bilevel optimization attack, on the other hand, drives the shared variables to the target values within two iterations of when the attacker initiates the attack. Thus, the attacker can select when the algorithm terminates.

C. Detection Results

We detect attacks using NNs trained on the shared variable mismatches, which follow different patterns when an attacker manipulates the shared data. The mismatches for algorithms run with no attack, PID attack, and bilevel attack on the IEEE 118-bus case are plotted in Figure 5. In this simulation, agent 2 is the attacker and the plotted mismatches correspond to those from agent 1. The plot shows an attack that begins at iteration 100, at which point the mismatches for attacked algorithms diverge from non-attacked operation.

We perform detection in a distributed manner using only local information. Each local agent trains a NN to classify the results of a distributed OPF problem as valid (not attacked) or invalid (attacked). To generate data, we run the distributed OPF algorithm repeatedly on the IEEE 118-bus case under no attack, PID attack, and bilevel attack. With each run, we randomly perturb the loads by 50% to 150% of the nominal values with a uniform distribution. The iteration at which the attack begins is randomly selected from the interval $[50, 100]$. In addition, for the PID attack strategy, we randomly select parameters k_p, k_d from the interval $(0, 0.4)$ with a uniform distribution, and fix $k_i = 0.001$. We create and train the NN using Flux [25]. The detection process operates on the last 50 shared variable mismatch values prior to the algorithm's termination. We train separate NNs for each attack type. For each attack strategy, we use 16000 shared variable mismatch vectors for training and reserve 4000 for testing.

The trained NNs detect attacks with 100% accuracy on the test data; that is, each of the 4000 test mismatch sequences are classified correctly. As seen in Figure 5, the shared variable mismatches behave quite differently under PID or bilevel attack, so classification is relatively straightforward. However, there may be other attack strategies which do not exhibit such obviously different convergence patterns and could avoid detection by our NNs. Note that the bilevel attack proposed in this paper likely would not be detected by the information estimation algorithm proposed in [13–15], as it converges too quickly for the defender to make predictions on historical data. This motivates future work to explore new attack strategies and determine how well our detection mechanism generalizes to attacks which it was not trained on.

VI. Conclusion

Distributed optimization algorithms have many desirable features for future electric power systems. However, their reliance on the communication infrastructure increases their vulnerability to cyberattacks. This paper has presented three cyberattack threat models that manipulate the data shared among the agents in order to change the solution of the OPF problem. We simulated these cyberattacks on the APP algorithm with three test cases. We considered a scenario where the attacker exploits the distributed algorithm to increase the total generation in one region of the system. We also demonstrate how a NN trained to recognize convergence patterns in the shared data can detect attacks with high accuracy.

In our ongoing work, we are evaluating the scalability of the attack strategies considered in the paper to larger systems with more agents. In addition, we are exploring new attack strategies which might bypass our detection mechanism. We are also extending these results to additional power flow models and considering alternative threat models where the attacker has only partial knowledge of the OPF problem and the agents' states.

References

[1] "Executive Order 13800, Strengthening the Cybersecurity of Federal Networks and Critical Infrastructure, Section 2(e): Assessment of Electricity Disruption Incident Response Capabilities," 2017.

[2] Z. El Mrabet *et al.*, "Cyber-security in smart grid: Survey and challenges," *Computers & Electrical Engineering*, vol. 67, pp. 469–482, 2018.

[3] A. S. Musleh, G. Chen, and Z. Y. Dong, "A survey on the detection algorithms for false data injection attacks in smart grids," *IEEE Trans. Smart Grid*, vol. 11, no. 3, pp. 2218–2234, 2020.

[4] D. K. Molzahn, F. Dörfler, H. Sandberg, S. H. Low, S. Chakrabarti, R. Baldick, and J. Lavaei, "A survey of distributed optimization and control algorithms for electric power systems," *IEEE Trans. Smart Grid*, vol. 8, no. 6, pp. 2941–2962, 2017.

[5] A. Kargarian, M. Mehrtash, and B. Falahati, "Decentralized implementation of unit commitment with analytical target cascading: A parallel approach," *IEEE Trans. Power Syst.*, vol. 33, no. 4, pp. 3981–3993, 2017.

[6] M. Alkhraijah, C. Menendez, and D. K. Molzahn, "Assessing the impacts of nonideal communications on distributed optimal power flow algorithms," to appear in *Electric Power Syst. Res.,* presented at *22nd Power Syst. Comput. Conf. (PSCC)*, 2022.

[7] S. Sridhar and M. Govindarasu, "Model-based attack detection and mitigation for automatic generation control," *IEEE Trans. Smart Grid*, vol. 5, no. 2, pp. 580–591, 2014.

[8] Q. Zhou, M. Shahidehpour, A. Alabdulwahab, and A. Abusorrah, "A cyber-attack resilient distributed control strategy in islanded microgrids," *IEEE Trans. Smart Grid*, vol. 11, no. 5, pp. 3690–3701, 2020.

[9] M. H. Cintuglu and D. Ishchenko, "Secure distributed state estimation for networked microgrids," *IEEE Internet of Things J.*, vol. 6, no. 5, pp. 8046–8055, 2019.

[10] J. Shi, S. Liu, B. Chen, and L. Yu, "Distributed data-driven intrusion detection for sparse stealthy FDI attacks in smart grids," *IEEE Trans. Circuits Syst. II: Express Briefs*, vol. 68, no. 3, pp. 993–997, 2021.

[11] C. Zhao, J. He, P. Cheng, and J. Chen, "Analysis of consensus-based distributed economic dispatch under stealthy attacks," *IEEE Trans. Ind. Electron.*, vol. 64, no. 6, pp. 5107–5117, 2017.

[12] J. Duan and M.-Y. Chow, "A resilient consensus-based distributed energy management algorithm against data integrity attacks," *IEEE Trans. Smart Grid*, vol. 10, no. 5, pp. 4729–4740, 2019.

[13] J. Duan, W. Zeng, and M.-Y. Chow, "Attack detection and mitigation for resilient distributed DC optimal power flow in the IoT environment," in *IEEE Int. Symp. Ind. Electronics*, 2016, pp. 606–611.

[14] ——, "An attack-resilient distributed DC optimal power flow algorithm via neighborhood monitoring," in *IEEE Power and Energy Society General Meeting*, 2016.

[15] ——, "Resilient distributed DC optimal power flow against data integrity attack," *IEEE Trans. Smart Grid*, vol. 9, no. 4, pp. 3543–3552, 2018.

[16] R. Harris, M. Alkhraijah, D. Huggins, and D. K. Molzahn, "On the impacts of different consistency constraint formulations for distributed optimal power flow," in *Texas Power and Energy Conference*, 2022.

[17] B. Stott, J. Jardim, and O. Alsaç, "DC power flow revisited," *IEEE Trans. Power Syst.*, vol. 24, no. 3, pp. 1290–1300, 2009.

[18] B. H. Kim and R. Baldick, "Coarse-grained distributed optimal power flow," *IEEE Trans. Power Syst.*, vol. 12, no. 2, pp. 932–939, 1997.

[19] J. Fortuny-Amat and B. McCarl, "A representation and economic interpretation of a two-level programming problem," *J. Oper. Res. Soc.*, vol. 32, no. 9, pp. 783–792, 1981.

[20] S. Babaeinejadsarookolaee *et al.*, "The power grid library for benchmarking ac optimal power flow algorithms," *arXiv:1908.02788*, 2019.

[21] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Rev.*, vol. 59, no. 1, pp. 65–98, 2017.

[22] I. Dunning, J. Huchette, and M. Lubin, "Jump: A modeling language for mathematical optimization," *SIAM Review*, vol. 59, no. 2, pp. 295–320, 2017.

[23] J. D. Garcia, G. Bodin, and A. Street, "Bileveljump.jl: Modeling and solving bilevel optimization in julia," *arXiv:2205.02307*, 2022.

[24] C. Coffrin, R. Bent, K. Sundar, Y. Ng, and M. Lubin, "Powermodels.jl: An open-source framework for exploring power flow formulations," in *2018 Power Systems Computation Conference*, June 2018, pp. 1–8.

[25] M. Innes *et al.*, "Fashionable modelling with Flux," *arXiv:1811.01457*, 2018.