Analyzing Malicious Data Injection Attacks on Distributed Optimal Power Flow Algorithms

Mohannad Alkhraijah,* Samuel Litchfield,† Craig Raslawski,† 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 DC optimal power flow (DC OPF) problem with and without manipulating the shared data and discuss the detectability of these attacks.**

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

I. Introduction

Power systems are transitioning from centralized optimization and control towards a more decentralized paradigm. With rapid growth in distributed energy resources (DER) 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. Coordinating DERs and multiple active end-users poses many challenges to system operators in terms of computational complexity and data sharing. The power systems research community has started developing a new paradigm that operates the power system in a more decentralized manner using distributed algorithms [4].

Distributed algorithms provide many desirable features for solving optimization problems for systems with massive numbers of DERs. This paper specifically considers the optimal power flow (OPF) problems that are routinely solved to optimally dispatch generators. In distributed OPF algorithms, agents compute optimal setpoints for their region of the power system by sharing the computational task with their neighboring agents rather than using a centralized entity to coordinate their operation. Although distributed algorithms can potentially increase the resiliency of power system operations by avoiding the existence of a 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 the convergence of distributed algorithms to the optimal solution [5, 6]. The impact and detection of false data injection attacks on distributed optimization algorithms has been discussed in the literature 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 the data communicated between agents in distributed algorithms for solving OPF problems.

To the best of our knowledge, the vulnerabilities of distributed DC OPF solution algorithms were first investigated in [13]. The authors of [13] show that an attacker can manipulate the results of distributed DC OPF based on the primal-dual gradient descent method by simply sending values for the shared variables corresponding to the attacker's target operating point. References [14–16] also investigate false data injection attacks on distributed DC OPF algorithms based on primal-dual gradient descent and propose a detection method. The detection method in [14–16] uses information estimation based on data from the previous iterations obtained from two-hop neighbors, i.e., neighbors of the neighbor, to detect incorrect information shared by neighboring agents. Further, the authors of [14–16] proposed using a *reputation index* to track the agents' historical behavior and identify malicious agents when the reputation index exceeds a certain threshold.

In all these references, the authors consider a simplistic attack scenario where the attacker sends values for the shared variables corresponding to the attacker's target operating point during all iterations. A more realistic scenario could involve an attacker that uses a more sophisticated manipulation scheme to mislead the other agents. For example, an attacker might initiate an attack in the first iteration which will make estimating of the correct response difficult. Further, the reputation index scheme could fail to detect the manipulation if the attack drives the distributed algorithm to terminate after a small number of iterations.

False data injection attacks may impact the convergence of distributed OPF algorithms in many ways depending on the attacker's objective. For instance, by sending random data that significantly deviates from what would be produced by the distributed algorithm, an attacker may prevent the algorithm

from reaching consensus on the shared variables or drive the solution to an infeasible operating point [6]. However, such attacks may be noticeable by the other agents and easily detected [11]. A more interesting case is to find an attack strategy that drives the distributed algorithm's solution to the attacker's target while considering the local constraints of the other agents.

In this paper, we investigate the vulnerability of the Auxiliary Problem Principle (APP) algorithm to adversarial attacks on the shared data. After describing a threat model, we propose three cyberattack strategies 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.

The main contribution of the paper is proposing and analyzing different cyberattack strategies that manipulate the distributed algorithm's solution. By using the PID feedback based strategy, we show that an attacker can drive the solution of the distributed OPF algorithm to their malicious target in fewer iterations compared to just sending the target values. This attack can also be more difficult to detect. Additionally, we propose a bilevel optimization problem that can be used by an attacker to drive the distributed algorithm's solution to a malicious target solution in one iteration.

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

II. Problem Formulation

The optimal power flow (OPF) problem is fundamental to the operation of electric power systems. OPF problems search for operating points that optimize the operation of electric power systems while satisfying constraints from both the network model (the power flow equations) and engineering limits on line flows, generator outputs, etc. The non-linear power flow equations induce non-convexities in the OPF problem. Since the convergence guarantee of the APP algorithm is limited to convex problems, we use the linear DC power flow approximation [17] to formulate the OPF problem (DC OPF). By avoiding the possibility of APP convergence failures due to nonconvexity, the DC OPF problem provides a useful starting point to investigate the vulnerabilities of distributed algorithms. 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

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

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

subject to:

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

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

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

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 $(i,j) \in \mathcal{L}$. The state of the buses is defined 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

Distributed optimization algorithms require the agents managing interconnected regions to communicate in order to share data with their neighbors. In this section, we describe the Auxiliary Problem Principle (APP) distributed algorithm that we use to solve the DC OPF problem and indicate how the agents share their computations. We consider the APP algorithm since numerical results indicate that it is more robust than other distributed algorithms to random communication errors when solving DC OPF problems [6].

To solve the OPF problem using the APP algorithm, we need to decompose the power system into regions. Each region is assumed to be 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 an Augmented Lagrangian technique and applying the APP algorithm, the DC OPF problem can be iteratively solved using a sequence

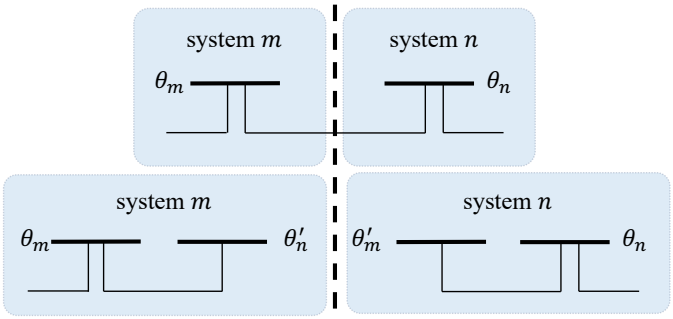

Fig. 1. Illustrative example of the network decomposition.

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. Thus, each problem is solved by each agent independently from other agents. The local problem corresponding to region m at iteration $k+1$ is:

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

subject to:

$$p_i^{k+1} - d_i = \sum_{(i,j)\in\mathcal{L}_m} B_{ij}(\theta_i^{k+1} - \theta_j^{k+1}), \qquad \forall i \in \mathcal{N}_m \quad (2b)$$
$$P_g^{min} \le p_g^{k+1} \le P_g^{max}, \qquad \forall g \in \mathcal{G}_m \quad (2c)$$
$$-P_{ij}^{max} \le B_{ij}(\theta_i^{k+1} - \theta_j^{k+1}) \le P_{ij}^{max}, \quad \forall (i,j) \in \mathcal{L}_m \quad (2d)$$

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 cardinally similar to $\mathcal{N}_m^s$, that denote the shared variables' values obtained by solving 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):

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

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 is capable of generating arbitrary shared variable values during each iteration of the APP algorithm independently of the physics of the underlying power system. The attacker also 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, we consider an attacker that seeks to manipulate the collective convergence of the APP algorithm such that the algorithm converges to a solution where other regions buy more power from the attacker's region relative to the true optimal solution, thus increasing the revenue for the attacker's region. However, we again note that this objective could be replaced by other goals corresponding to different attacker motivations.

In order to achieve their goal, the attacker must find an advantageous convergent state for the system and then determine an approach for driving the shared variable values such that the APP algorithm converges to that state. We propose several 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. The remainder of this section describes the three attacker strategies that we analyze in this paper.

1) Simple Attack: In the simplest case, the attacker determines an advantageous convergent state by modeling the system-wide OPF problem, but modifying it by manipulating the generation cost functions. This results in a solution where neighboring regions purchase more power from the region controlled by the attacker. 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. In other words, these variables do not change during execution of the algorithm. This is the least complex approach, but also the least subtle, increasing the likelihood of detection.

2) Feedback Controller: To make their manipulations of the shared variables less obviously detectable than the simple attack strategy of sending the same values at every iteration, we next consider an attack strategy that incorporates a feedback control mechanism. In this attack strategy, 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.

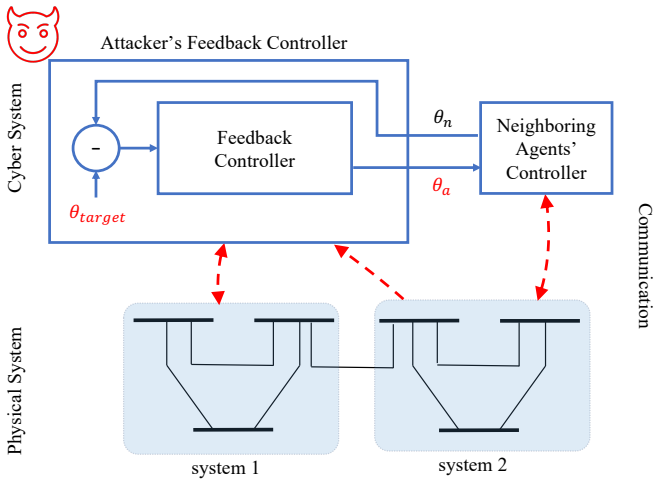

Fig. 2. Feedback control attack model.

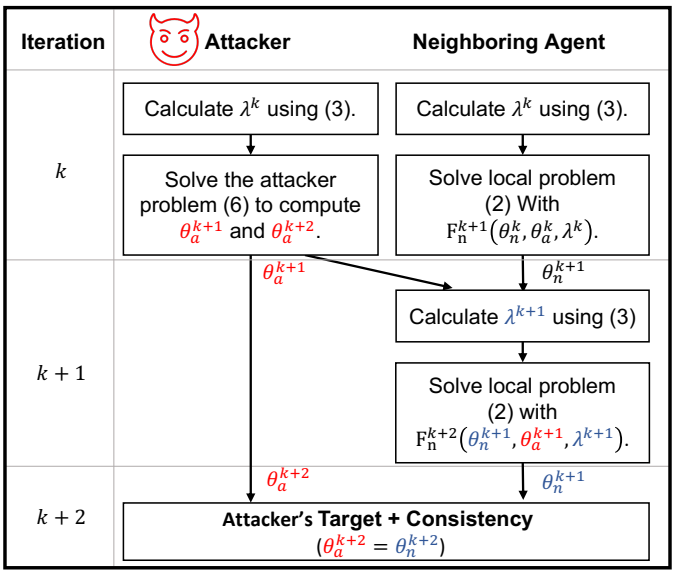

Fig. 3. Flow diagram visualizing the bilevel attack strategy.

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:

$$e_k = \theta_{target} - \theta_n^k. \tag{4}$$

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) in order to obtain the attacker's shared variable values for the next iteration, θ_a^{k+1}:

$$\theta_a^{k+1} = \theta_{target} + K_p e_k + K_i \sum_{n=0}^{k} e_n + K_d(e_k - e_{k-1}). \tag{5}$$

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: This model involves solving a bilevel optimization problem with a target objective in the upper level and the neighboring agents' problems in the lower level. The target objective in the upper level is selected by the attacker to gain an advantage over the other agents. The lower level consists of the neighboring agents' problems to ensure that the bilevel optimization solution is always optimal with respect to the neighboring agents. 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 and system specific characteristics such as the network structure, electrical parameters, etc.

To successfully compute an attack to 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:

$$\min \quad G_{target}(p_a^{k+2}, \theta_a^{k+2}) \tag{6a}$$

subject to:

$$(p_n^{k+1}, \theta_n^{k+1}) = \underset{(p_n^{k+1}, \theta_n^{k+1}) \in A_n}{\arg\min} \quad F_n^{k+1}(\theta_n^k, \theta_a^k, \lambda^k) \tag{6b}$$

$$(p_n^{k+2}, \theta_n^{k+2}) = \underset{(p_n^{k+2}, \theta_n^{k+2}) \in A_n}{\arg\min} \quad F_n^{k+2}(\theta_n^{k+1}, \theta_a^{k+1}, \lambda^{k+1}) \tag{6c}$$

$$(p_a^{k+2}, \theta_a^{k+2}) \in A_a \tag{6d}$$

$$\theta_a^{k+2} = \theta_n^{k+2} \tag{6e}$$

$$\lambda^{k+1} = \lambda^k + \alpha(\theta_n^{k+1} - \theta_a^{k+1}) \tag{6f}$$

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 (6b) and (6c) 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 (6b) where the inputs of F_n^{k+1} are constants, the inputs of the objective function F_n^{k+2} in (6c) contain decision variables. Both of the objective functions of the lower-level problem 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 (6) to simplify the representation of the attacker's problem. However, the attacker solves the first iteration of the lower-level problem (6b) before solving the bilevel problem. We

solve the bilevel problem by reformulating the second iteration lower-level problem (6c) using its Karush–Kuhn–Tucker (KKT) conditions parametrized with the upper-level variables, i.e., θ_a^{k+1}. The parametrized 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 and 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 will drive the neighboring agents' solution and change 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. 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 MATPOWER [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 second agent solves 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 model 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 on MATLAB using a PC with an Intel Core i7 processor and 16 GB of RAM. We use Gurobi to solve all optimization problems, and we use YALMIP [21] to reformulate the bilevel optimization problems (6) as a tractable MILP [21].

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 MATPOWER 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 all successfully drive the results of the APP algorithm to the attacker's target values. We use the *optimility gap*, 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. Figures 4–6 show the convergence of the three test cases to a suboptimal solution using the proposed attack

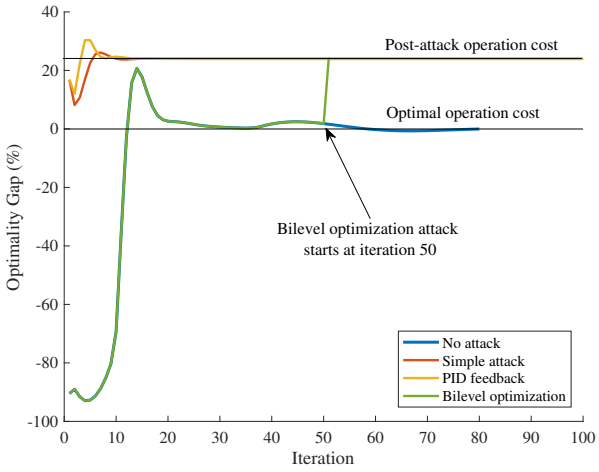

Fig. 4. 14-Bus relative cost convergence pre- and post-attack.

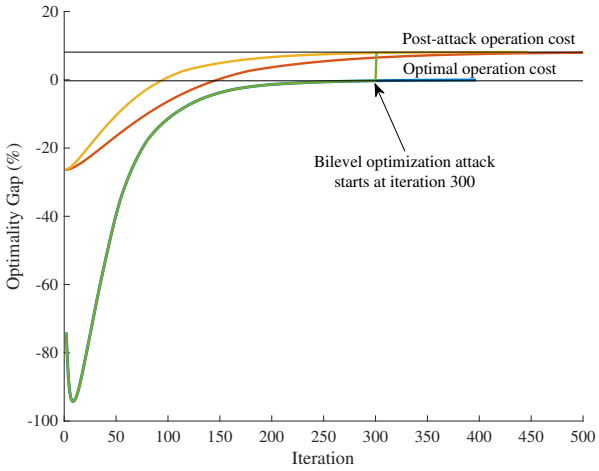

Fig. 5. 39-Bus relative cost convergence pre- and post-attack.

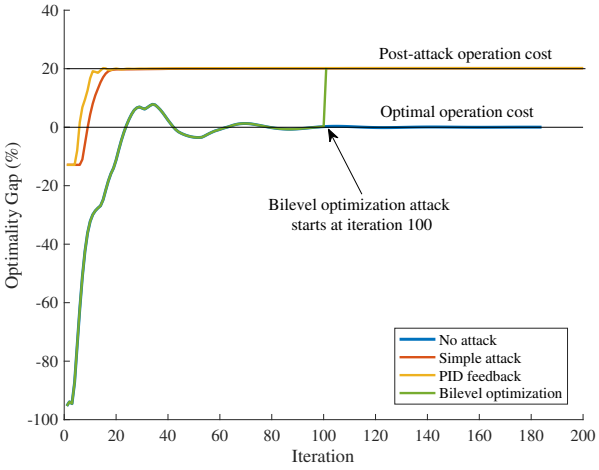

Fig. 6. 118-Bus relative cost convergence pre- and post-attack.

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 one iteration of starting the attack, the timing of which can be selected by the attacker.

strategies. The figures show the operation cost of the optimal solution when there is no attack 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 Figs. 4–6. However, for both of these attacks, it takes a larger number of iterations for the 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 simple attack takes 1409, 648, and 3735 iterations to converge compared to 981, 447, and 2778 iterations for the PID feedback 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 one iteration of when the attacker initiates the attack. Thus, the attacker can select when the algorithm terminates.

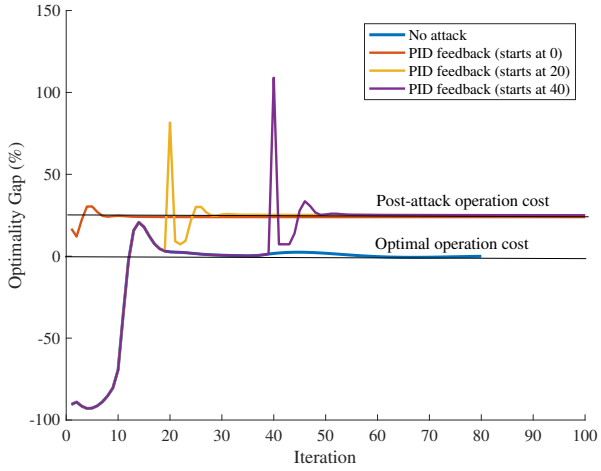

Fig. 7. Convergence of the cost prior to and after initiating the feedback control attack strategy.

C. Discussion on Detection Mechanisms

The simulation results of the three attack strategies show the vulnerability of the APP algorithm to cyberattacks. For each strategy, the attacker determines shared variable values that drive the results of the distributed algorithm to the target values. This indicates the importance of having detection mechanisms against data manipulation. The detection mechanism should be capable of differentiating between shared variable values from typical (non-attacked) distributed optimization algorithms and manipulated values.

For each of the three attack strategies considered in this paper, we observe a sudden change in the convergence pattern in the presence of data manipulation. For instance, as shown in Fig. 7, the feedback control attack strategy causes a large spike in the relative cost in the iteration after the attack commences. We observe similar behavior with the simple attack strategy.

The values of the Lagrange multipliers provide another indication of manipulations to the shared variable values. These values suddenly change once the attacker initiates the bilevel optimization attack as shown in Fig. 8. The users of

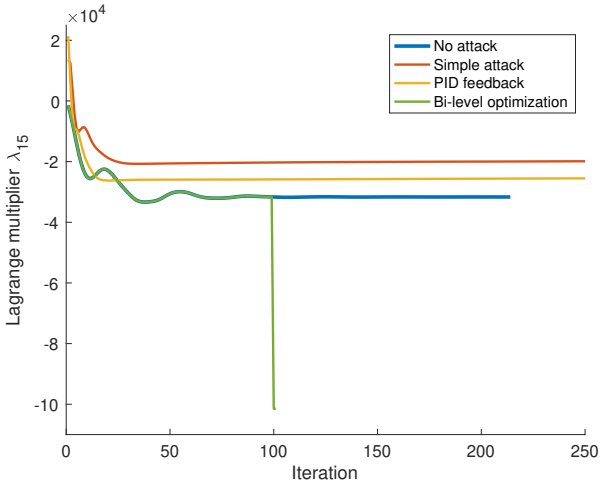

Fig. 8. Lagrange multiplier convergence pre- and post-attack.

the distributed algorithms could consider those abnormalities as sign of potential data manipulation. Our ongoing work is developing more rigorous detection methods to identify

and characterize potential cyberattacks to the shared variable values in distributed optimization algorithms.

V. 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 a cyberattack threat model and attack strategies 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.

In this paper, we briefly discussed impacts from the data manipulation that appear distinguishable from the typical convergence behavior of the APP algorithm. In our ongoing work, we are leveraging these observed impacts to develop pattern recognition methods capable of reliably detecting data manipulation. Moreover, we are evaluating the scalability of the attack strategies considered in the paper to larger systems with more agents. We are also 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," August 2017.

[2] Z. El Mrabet, N. Kaabouch, H. El Ghazi, and H. El Ghazi, "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 Transactions on 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 Transactions on 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 Transactions on Power Systems*, vol. 33, no. 4, pp. 3981–3993, 2017.

[6] M. Alkhraijah, C. Menendez, and D. K. Molzahn, "Evaluating the performance of distributed optimal power flow algorithms with nonideal communication," *arXiv:2106.00135*, 2021.

[7] S. Sridhar and M. Govindarasu, "Model-based attack detection and mitigation for automatic generation control," *IEEE Transactions on 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 Transactions on 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 Journal*, 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 Transactions on Circuits and Systems 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 Transactions on Industrial Electronics*, 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 Transactions on Smart Grid*, vol. 10, no. 5, pp. 4729–4740, 2019.

[13] J. Duan, W. Zeng, and M.-Y. Chow, "Economic impact of data integrity attacks on distributed DC optimal power flow algorithm," in *North American Power Symposium (NAPS)*, 2015, pp. 1–7.

[14] ——, "Attack detection and mitigation for resilient distributed DC optimal power flow in the IoT environment," in *2016 IEEE 25th International Symposium on Industrial Electronics (ISIE)*, 2016, pp. 606–611.

[15] ——, "An attack-resilient distributed DC optimal power flow algorithm via neighborhood monitoring," in *IEEE Power and Energy Society General Meeting (PESGM)*, 2016, pp. 1–5.

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

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

[18] B. H. Kim and R. Baldick, "Coarse-grained distributed optimal power flow," *IEEE Transactions on Power Systems*, 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," *Journal of the Operational Research Society*, vol. 32, no. 9, pp. 783–792, 1981.

[20] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–19, 2011.

[21] J. Löfberg, "YALMIP: A Toolbox for Modeling and Optimization in MATLAB," in *IEEE International Symposium on Computer Aided Control Systems Design (CACSD)*, September 2004, pp. 284–289.