Convergence and Recovery Analysis of the Secure Distributed Control Methodology for D-NCS

Wente Zeng, *Student Member, IEEE*, Mo-Yuen Chow, *Fellow, IEEE*
Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC, USA
wzeng3@ncsu.edu, chow@ncsu.edu

***Abstract*—Distributed control algorithms (e.g., consensus algorithms) are vulnerable to the misbehaving agent compromised by cyber attacks in the Distributed Networked Control Systems (D-NCS). In this paper we continue our work on a proposed secure distributed control methodology that is capable of performing a secure consensus computation in D-NCS in the presence of misbehaving agents. The methodology is introduced first and proven to be effective through convergence analysis. Then, we extend our secure distributed control methodology to the leaderless consensus network by introducing and adding two recovery schemes into the current secure distributed control framework to guarantee accurate convergence in the presence of misbehaving agents. All phases of our method are distributed. That is, at each step of the detection, mitigation, identification, update, and recovery, each agent only uses local and one-hop neighbors' information. The simulation results are presented to demonstrate the effectiveness of the proposed methods.**

I. Introduction

Distributed Networked Control Systems (D-NCS) have been at the core of national critical infrastructures and industrial control systems for many decades (e.g., electrical power systems and transportation systems). While most D-NCS have been safe in the past, a few confirmed cases of cyber attacks have occurred [1]. The recent presence of the infamous industrial control system malwares "Stuxnet" and "Flame" have brought significant attention to making industrial control systems safe from such malicious cyber attacks [2]. Many D-NCS applications are time-sensitive, data-sensitive, and safety-critical. The potential consequences of compromising D-NCS can be devastating to public health and safety, national security, and the economy. Therefore, it is important to implement D-NCS with secure controls that make reliable, safe, and flexible performance possible.

D-NCS are increasingly more vulnerable to cyber attacks with the rapid advancements and uses of networking, embedded systems, wireless communication technologies, and novel control strategies [3]. In particular, more and more distributed control algorithms are being used in D-NCS because of their flexibility, robustness, computation, and communication features [4]. These algorithms, however, increase the vulnerability of D-NCS to malicious cyber attacks. In the absence of a centralized supervisory node that monitors the activities of the nodes in the network, distributed control strategies are prone to cyber attacks and component failures. Thus, it is increasingly important to guarantee that computations are secure and trustworthy even in the presence of misbehaving devices. Also, most of the current efforts for protecting D-NCS have been accomplished by prevention and are limited to communication security [5, 6]. There is an urgent growing need to protect control algorithms from malicious cyber attack.

One typical task in the NCS is to agree upon a certain performance measure for a group of agents, such as the work load on a network of parallel computers, the clock speed for wireless sensor networks, or the velocity or formation pattern for a group of autonomous vehicles [7, 8]. Several distributed control algorithms, such as consensus algorithms and gossip algorithms, have been proposed and studied in D-NCS to accomplish such tasks (e.g., formation control of multi-robot systems, time synchronization of wireless sensor networks) [4]. We discuss the linear consensus algorithm in this paper. In this algorithm, the state of each node is updated at each time step, which is a weighted average of its own state and those received from its neighbor nodes [9]. All the nodes are assumed to cooperate and follow the protocol exactly; otherwise, the consensus would not be guaranteed to be reached. Thus, it is important to ensure secure computation in the face of failures of and intrusions into the linear consensus algorithm.

In the computer science community, the security problems of consensus protocols, such as the famous Byzantine Generals Problem, have been extensively studied in distributed computing [10]. That problem deals with the resilience of distributed systems to malicious (Byzantine) nodes. It has been shown that the well-behaving nodes of a network can always agree upon a certain value if, and only if, the number of malicious nodes is: 1) less than 1/2 of the network connectivity, and 2) less than 1/3 of the total number of nodes in the network [11]. This result has been regarded as a fundamental constraint of the distributed consensus protocol to sustain malicious nodes in the network, and many relevant Byzantine Fault Tolerance (BFT) algorithms have been proposed [12]. However, the continuity and local dynamics of the individual agent, which are prominent in control system applications, haven't been considered and discussed in these approaches.

Research on the security issues of distributed control algorithms is still in its infancy. In the literature several results can be found, most of which deal with the security issues of the linear consensus algorithm from a control theory perspective. Pasqualetti *et al.* [8] first introduced the problem of detecting and identifying a single misbehaving

agent in a linear consensus network and an observer-based solution was proposed. Sundaram and Hadjicostis [13] addressed the resilience property of the linear consensus algorithm by studying the relationship between the number of the malicious nodes and the connectivity of the network. Teixeira *et al.* [14] proposed a distributed scheme to detect and isolate the cyber attacks on the communication network of a D-NCS using observers and discussed how to reduce the number of observer nodes while maintaining the coverage of the entire network. In most of the above methods, all the agents need to have full knowledge of the topology of the entire network (i.e., global information) in a centralized manner. Furthermore, the computational overhead for these methods is generally quite high, and the intrusion detection schemes and control algorithms are always separated. There is no feedback to the control parameters when compromised nodes are detected or identified. Instead, our objective is to develop secure distributed control algorithms that have been set up with an eye toward quality and security design in the first place, so they are more likely to remain secure and less vulnerable to hacking, intrusion, or malicious control.

In our previous work [15], several fundamental issues of reaching consensus with a group of agents via distributed computation in the presence of one misbehaving agent were investigated, and a secure distributed control methodology that embeds a built-in defense mechanism for the leader-follower consensus network was proposed. As regards our research in this direction, the convergence analysis of the secure distributed control methodology is discussed in this paper. We then extend our secure distributed control methodology to the leaderless consensus network by introducing and adding two recovery schemes into the current secure distributed control framework. This guarantees an accurate convergence in the presence of misbehaving agents.

The remaining sections are organized as follows: Section II describes the secure distributed control methodology. Section III provides the convergence proof and illustrates the constraints and limitations of the proposed method through convergence analysis. Section IV introduces two recovery schemes that are added to ensure correct convergence of the secure distributed control methodology in the presence of misbehaving agents in leaderless consensus networks. Section V demonstrates the effectiveness of the proposed techniques through simulation, and Section VI concludes this paper.

II. Secure Distributed Control Methodology

Let us consider a network of n agents (e.g., robots) whose identical linear dynamics (e.g., first order or second order dynamics) are denoted by:

$$x_i[k+1] = Ax_i[k] + Bu_i[k], \tag{1}$$

where x_i and u_i are the agent states and controls and i is the index for the agents in the network [16].

Each agent receives the following measurements:

$$y_i[k] = Cx_i[k] \tag{2}$$

and implements a reference-based proportional controller on-board:

$$u_i[k] = K_p e_i[k] = K_p(r_i[k] - y_i[k]), \tag{3}$$

where r_i is the reference state and K_p is the proportional gain for the P controller to be designed.

To reach consensus of these agents cooperatively, the agents communicate with one another through wireless communications. In this neighbor-to-neighbor information exchange process, the first-order linear consensus algorithm is implemented in the consensus manager of each agent. Therefore, at each time instant, each agent updates its reference state as a weighted combination of its own current state and other measured states received from its neighbors, as shown in Figure 1.

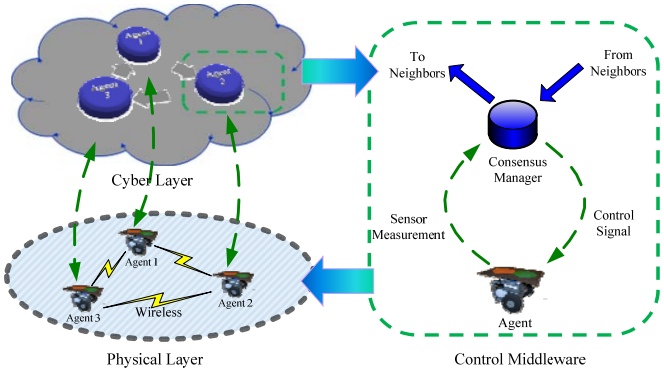

Fig. 1. The framework of multi-agent distributed control in D-NCS.

The reference state update rule in the consensus manager of each agent is listed below:

$$z_i(k) = y_i(k), \tag{4}$$

$$z_i[k+1] = \sum_{j=1}^{n} d_{ij} z_j[k], i = 1,...,n, \tag{5}$$

$$r_i(k+1) = z_i(k+1), \tag{6}$$

where d_{ij}, the consensus computation weight, is the (i, j) entry of the row-stochastic matrix D and z_i is the information transmitted in the communication layer.

By following the update rule described in (5), all the agents will converge to a common state asymptotically, and thus the consensus task will be completed in a distributed manner. The convergence rate is based on the topology of the system's communication network (graph G), which is the second smallest eigenvalue of the associated Laplacian matrix.

The proposed secure distributed control methodology, as shown in Figure 2, embeds the following four phases into each iteration of the consensus computation process (i.e., reference state update): 1) Detect the neighbors' misbehaviors relying only on each agent's local observations through the Neighborhood Monitor; 2) Adjust the consensus computation weights according to the neighbors' reputation values via the Local Reputation Manager; 3) Identify and isolate the compromised agent; and 4) Use the adjusted consensus computation weights to calculate the updated reference state

and ensure the convergence of the well-behaving agents. The detailed description of each phase is given as follows:

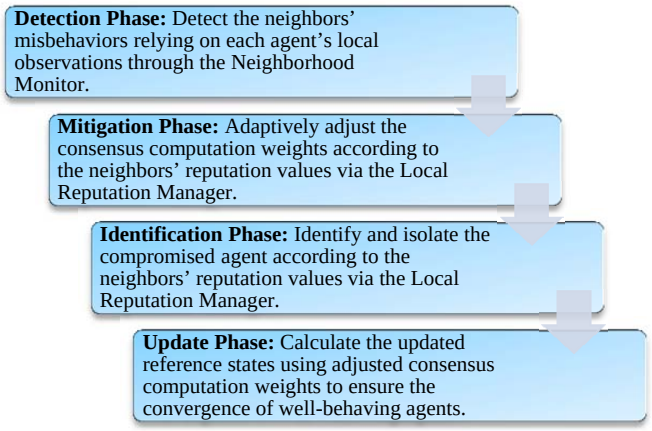

Fig. 2. Flowchart of the secure distributed control methodology.

A. Detection Phase

Definition (Neighborhood Monitor): A Neighborhood Monitor is an embeded monitor that enables an agent to observe the behaviors of its neighbors.

In this phase, with the assumption that listening to a neighbor's transmission is possible due to the characteristics of wireless communications, each agent constructs a Neighborhood Monitor to observe its neighbors and learn from them by eavesdropping on the transmissions of the one-hop neighbors within its communication range.

In this Neighborhood Monitor, the agent carries out a real-time anomaly detection mechanism for all its neighbors (similar to the watchdog design in the security mechanism of *ad hoc* networks). The agent redundantly calculates and stores its neighbor agent *j*'s reference state r_j and compares it with the state value z_j received from neighbor agent *j* in time step *k*. If $r_j[k] - z_j[k] \le \gamma(k,\delta)$, the neighbor is normal; if $r_j[k] - z_j[k] > \gamma(k,\delta)$, the neighbor has a potential anomaly.

$$G_{ij}[k] = \begin{cases} G_{ij}[k-1]+1, & r_j[k]-z_j[k] \le \gamma(k,\delta) \\ G_{ij}[k-1], & r_j[k]-z_j[k] > \gamma(k,\delta) \end{cases}, \tag{7}$$

where G_{ij} is the total number of verifiably correct agent states of neighbor agent *j* up to time step *k* monitored by agent *i*. $\gamma(k,\delta)$ is a threshold function that depends on the time step and the disturbance δ.

B. Mitigation Phase

Definition (Reputation Manager): A Reputation Manager is an onboard system for an agent that updates the reputation values of the neighbor agents and records them in its local reputation table.

Reputation is an index for the credibility of a node in the network; it is widely used in the cooperation issues among the nodes of an *ad hoc* network. Here, we introduce the reputation metric to quantitatively measure the credibility of the neighbor agents. If misbehaviors of one neighbor agent are detected, the Neighborhood Monitor then reports to the local Reputation Manager. We use the Bayesian Reputation function to calculate the reputation values [17, 18]. Given a set of verifiably correct and incorrect behaviors from a neighbor agent, the probability distribution of seeing a particular combination of correct and incorrect behaviors from this neighbor agent satisfies a *beta* distribution [18]. The expected value of the *beta* distribution forms the reputation. This value is given by the ratio of the number of correct agent states received to the number of total agent states received:

$$rep_{ij}[k] = \frac{\eta G_{ij}[k]+1}{\eta k + 2}, \tag{8}$$

where rep_{ij} is the reputation value of neighbor agent *j* up to time step *k* in agent *i*'s local Reputation Manager. η is the reputation coefficient that can adjust the changing speed of the reputation value regarding to different applications.

In the mitigation phase, if the result of the Neighborhood Monitor is a misbehavior or good behavior, it is used to update the neighbor agent's reputation value in each agent's local Reputation Manager.

C. Identification Phase

If the reputation value falls below a certain level (i.e., malicious threshold rep_{th}), the corresponding neighbor agent is identified as a compromised agent and is then isolated. In this case, all the information from the compromised neighbor agent is rejected. Nevertheless, a timeout mechanism may be used to allow the compromised agent to rejoin the network if it has been falsely accused in the past or if it behaves normally again. The Reputation Manager acts as a confirmation mechanism that one agent is confirmed as misbehaving if it is detected with abnormal values repeatedly during a certain period of time. If this occurs, it is interpreted as the agent's reputation value dropping below the malicious threshold.

For any warning and intrusion detection mechanism, there is essentially a trade-off between the false reject rate (FRR) and the false accept rate (FAR). In our methodology, the design preferences are embodied in the threshold functions and in the reputation coefficient η.

D. Update Phase

In order to embed the above security mechanisms in the consensus computation process, an update phase is proposed to adaptively update the consensus computation weights d_{ij}, which is the (*i*, *j*) entry of the row-stochastic matrix *D*, based on the reputation values from the local Reputation Manager. The method is shown as:

$$d_{ij}[k] = rep_{ij}[k] \Big/ \sum_{j=1}^{n} rep_{ij}[k]. \tag{9}$$

Thus, the reference state update rule in the consensus manager is correspondingly changed to:

$$z_i[k+1] = \sum_{j=1}^{n} d_{ij}[k] z_j[k], i = 1,\ldots,n. \tag{10}$$

In this secure distributed control methodology, if the neighbor agent's reputation drops, the consensus manager will gradually decrease the corresponding consensus computation weights, d_{ij},

to slow down the speed at which the malicious effects of that potentially compromised agent spreads. If a neighbor agent is identified as a compromised node, the consensus manager will set the corresponding d_{ij} to zero to cut off the connection. Since every agent performs this adaptive consensus computation in parallel, the well-behaving agents finally can isolate the compromised agent and converge to a steady state.

III. Convergence Analysis

Definition (Asymptotic Consensus): The system is said to reach asymptotic consensus if $|x_i[k] - x_j[k]| \to 0$ as $k \to \infty$, for all $i, j \in V$.

A. Convergence Proof

Lemma:

If the total number of misbehaving agents is less than the connectivity of the network, the entire network will not be disconnected due to the isolation of some critical nodes being compromised. In other words, if there is always a spanning tree associated with the updating matrix D, the consensus of the well-behaving nodes can be achieved asymptotically.

Proof:

First, we need the following lemmas from [9] and [19] to derive our proof.

Lemma 1:

The discrete linear consensus algorithm achieves asymptotic consensus if, and only if,

$$D[k-1]D[k-2]\cdots D[2]D[1]D[0] \to 1c^T. \qquad (11)$$

As $k \to \infty$, where **1** denotes the $n \times 1$ column vector with all the entries equal to 1 and c is an $n \times 1$ vector of constant coefficients.

Lemma 2:

A stochastic matrix P is called indecomposable and aperiodic (SIA) if $\lim_{n\to\infty} P^n = 1y^T$.

Lemma 3:

If the union of a set of directed graphs $\{G_1, G_2, \ldots, G_m\}$ has a spanning tree, then the matrix product $D_m D_{m-1} \ldots D_2 D_1$ is SIA, where D_i is a stochastic matrix with positive diagonal entries corresponding to each directed graph G_i.

In the proposed secure distributed control methodology, at each time step $k \in N$, each node communicates with its neighbors and updates its local value. This is achieved using equations (9) and (10), where $d_{ij}[k]$ is the weight assigned to node j's value by node i at time step k.

Since $d_{ij}[k] = \dfrac{rep_{th}}{\sum_{j=1}^{Vi} rep_{ij}[k]} \geq \dfrac{rep_{th}}{V_i}$,

where V_i is the set of neighboring nodes of the i^{th} node, we know there exists a constant $\alpha \in R$, $0 < \alpha < 1$ such that all of the following conditions hold:

- $d_{ii}[k] \geq \alpha, \forall i, k$
- $d_{ij}[k] = 0 \; if \; j \notin V_i[k], \forall i, j, k$
- $d_{ij}[k] \geq \alpha \; if \; j \in V_i[k], \forall i, j, k$
- $\sum_{j=1}^{n} d_{ij}[k] = 1, \forall i, k$

where $\alpha = \dfrac{rep_{th}}{V_i}$.

Thus, we know that $D[k]$ is a stochastic matrix with positive diagonal entries $\forall k$. The lower bound α on the weights is imposed to guarantee convergence.

Also, let us assume that the connectivity of the network is m. Since the total number of misbehaving agents is less than the connectivity of the network, the maximum number of malicious nodes is m-1. Even if all the m-1 malicious nodes are isolated in the worst case scenario, the entire network (except for the isolated misbehaving nodes) is still connected. Thus, there is still a spanning tree in the graph that is associated with the rest of the well-behaving nodes.

In conclusion, there is always a spanning tree in the graph containing the well-behaving nodes associated with the updating matrix D, which is a stochastic matrix with positive diagonal entries. According to lemmas 1, 2, and 3, all the well-behaving nodes (except for the isolated misbehaving nodes) will achieve consensus asymptotically.

B. Convergence and Robustness Analysis

Through observations, we discovered that if the misbehaving agent is detected and isolated, it may separate the network into two parts (in some certain network topologies), where there is no hope for any consensus algorithm to be working. Thus, before we remove the misbehaving agent from the network, we need to check the network's topology—more specifically, the connectivity of the network—to avoid a disconnection in the network. For example, if we identify one misbehaving agent in the network and we don't know the entire topology of the network, we need to make sure that the network is at least 2-connected before we isolate that agent.

To summarize, generally, our algorithm (or any secure control algorithm with an isolation scheme) works under the assumption that the total number of misbehaving agents is less than the connectivity of the network in the worst case. Otherwise, the entire network may be disconnected because of the isolation of some critical nodes that are being compromised. In general, all the security algorithms using isolation are worst-case bounded by the connectivity of the network. If the connectivity of the network is m, the maximum number of malicious nodes that these algorithms can tolerate is *m-1* in the worst case.

IV. Recovery Analysis

For a leader-follower consensus network, the task of security is simply to detect, identify, and isolate the misbehaving agents so that they have no impact on the well-behaving agents since all the well-behaving agents will eventually converge to the final state of the leader. In a leaderless consensus network, if we assume that we have been

able to isolate a misbehaving agent (using the proposed secure distributed control methodology), the well-behaving agents can still converge to a steady state. However, this state may be different from the desired final state (e.g., average value of the initial state of all the well-behaving agents). This is because the contribution of the misbehaving agent before isolation has already affected the consensus computation.

In order to eliminate the total contribution from of a misbehaving agent in a leaderless consensus network, rather than just isolating the misbehaving agent from the network to cancel out the misbehaving agent's effect on the network after isolation, we also want to compensate the agent's total effect from the time it was compromised onward until isolation. So, for the leaderless consensus network, we add a Recovery Phase to our proposed secure control methodology after the four regular phases. This fifth phase is to remove the impact of the misbehaving agents by applying recovery schemes.

We have developed two types of recovery schemes for different applications: rollback recovery and excitation recovery.

A. Rollback Recovery

The rollback recovery scheme lets all the agents periodically save their fault-free states (e.g., the initial states) as checkpoint states. When a misbehaving agent is identified and isolated, all the well-behaving agents will restore their states to the checkpoint states by rolling back to their fault-free states. Then, they will converge to the correct steady state through consensus computation. The rollback recovery scheme is suitable for static distributed control applications such as parallel computing and wireless sensor networks since the agent states are easy to reset in these cases.

B. Excitation Recovery

In dynamic distributed control scenarios such as multi-robot formation control, the agent states cannot be reset or rolled back because of the physical constraints. An intuitive solution is to apply extra excitations locally to each agent to compensate the compromised agents' effect from the time when they were compromised until isolation [20]. The excitation recovery scheme we proposed has the following steps:

1) At time $k = k_1$, all the agents start to record and update the information $\sum_{k=k_1}^{n} d_{ij}(k)[x_i(k) - x_j(k)]$ from all of their neighbors in every iteration.
2) At time $k = k_2$, the compromised agent is identified as compromised and isolated by the secure distributed control methodology. The neighboring agents of the compromised agent apply an external recovery control input $u_{rec}(k)$ such that from time $k = k_2$ to time $k = k_3$,

$$\sum_{k=k_2}^{k_3} u_{rec}(k) = -\sum_{k=k_1}^{k_2} d_{ij}(k)[x_i(k) - x_j(k)]. \qquad (12)$$

3) After a finite number of iterations at time $k = n$, a recovery of the correct weighted average of the states of the well-behaving agents will be performed if all the well-behaving agents are still connected.

As we can see, the accuracy of the final convergence value of this excitation recovery scheme depends on k_1 (when it starts to record the neighbors' information). If $k_1 = 0$, the convergence value will be the correct weighted average of the initial states of the well-behaving agents. However, each agent needs to record the contribution of both well-behaving and misbehaving agents before the misbehaver is detected, which would become inefficient and resource-consuming. If k_1 equals the time when the misbehaving agent is first compromised, then the final steady state will be the weighted average value with the well-behaving agents' initial states and the contribution of misbehaving agent before it is compromised. This will be slightly different from the accurate consensus value. Thus, there is a trade-off between convergence accuracy and algorithm efficiency. A different priority is preferred by setting an appropriate record starting time, k_1, for different scenarios/applications.

V. Simulation Results

To analyze the performance of the proposed secure distributed control methodology with recovery schemes, we apply rollback recovery and excitation recovery schemes to two different distributed control applications respectively: wireless sensor networks and multi-robot formation control.

A. Wireless Sensor Network

In this task, eight fully connected temperature sensors are used to measure the average temperature of an area. The readings of all the sensors are shown in Table I and let us assume Sensor 5 and Sensor 6 are malfunctioning under attack.

TABLE I: READINGS OF THE TEMPERATURE SENSORS

Sensor 1	*Sensor 2*	*Sensor 3*	*Sensor 4*	*Sensor 5*	*Sensor 6*	*Sensor 7*	*Sensor 8*
30	27	25	35	40	38	32	31

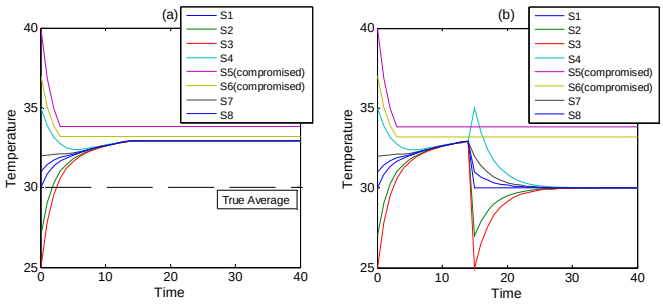

Fig. 3. The average consensus results using the proposed secure distributed control methodology in the presence of a malfunctioning sensor: (a) without rollback recovery; and (b) with rollback recovery.

Since the readings of the sensors are static, the rollback recovery scheme is used in this comparative case study. Figures 3(a) and 3(b) show the simulation results of the average consensus with two compromised sensors using the secure distributed control methodology without rollback recovery and with rollback recovery, respectively. As we can see, without a recovery scheme, the convergence value 32.9°C is different from the desired true average of the good sensor readings: 30°C

due to the contribution of two compromised sensors before isolation. In contrast, all the good sensors can finally converge to the correct average value by rolling back to their initial states after isolating the compromised sensors. This validates the effectiveness of the rollback recovery scheme.

B. Multi-Robot Formation Control

A representative formation control task, a 1-D rendezvous, is designed: Let all six robots stay in a line in parallel (reach the same x-coordinate which that is the average of their initial positions). Since leaderless consensus network is applied, the communications topology is set to be fully connected. Robot 5 is the misbehaving robot and is being compromised at $k = 5$. The initial positions of all the robots are shown in Table II.

TABLE II: INITIAL POSITIONS (x-COORDINATES) OF THE ROBOTS

Robot 1	*Robot 2*	*Robot 3*	*Robot 4*	*Robot 5*	*Robot 6*
120	0	80	10	100	50

Excitation recovery is used in this formation control case study, and the parameter settings are: $k_1 = 0$, $k_2 = 30$, and $k_3 = 35$. Let us carry out the rendezvous task using the proposed secure distributed control methodology to perform the consensus computation both with and without excitation recovery for comparison. As the simulation results shown in Figure 4, without excitation recovery, Robot 5 drags the entire team away from the desired rendezvous point even as it is detected and isolated from the network eventually. However, by applying the excitation recovery scheme, the neighboring robots apply the recovery input locally after disconnecting the compromised agent, and then all the well-behaving robots converge at the correct destination asymptotically. The convergence is guaranteed as long as the topology satisfies the sufficient conditions of the average consensus algorithm.

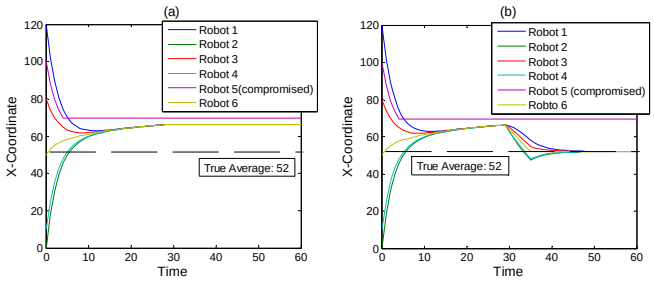

Fig. 4. The multi-robot formation control results using the proposed secure distributed control methodology in the presence of a compromised robot: (a) without excitation recovery; and (b) with excitation recovery.

VI. CONCLUSION

This paper discusses the secure distributed control problem in unreliable D-NCS. The proposed secure distributed control methodology is discussed and then proven to be effective through convergence analysis. Furthermore, we extend our secure distributed control methodology to the leaderless consensus network by adding two recovery schemes to guarantee accurate convergence. Applications to the wireless sensor networks and formation control are discussed, and their performance is verified through simulations.

ACKNOWLEDGEMENT

This research was supported under NSF-ECS-0823952 “Impaired Driver Electronic Assistant (IDEA)” project.

REFERENCES

[1] "Common Cyber Security Vulnerabilities Observed in Control System Assessments by the INL NSTB Program," U.S. Department of Energy Office of Electricity Delivery and Energy Reliability, 2008.

[2] R. Langner, "Stuxnet: Dissecting a Cyberwarfare Weapon," *Security & Privacy, IEEE*, vol. 9, pp. 49-51, 2011.

[3] A. Ardenas, S. Amin, and S. Sastry, "Research challenges for the security of control systems," in *Proceedings of the 3rd conference on Hot topics in security*, San Jose, CA, 2008, pp. 1-6.

[4] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and Cooperation in Networked Multi-Agent Systems," *Proceedings of the IEEE*, vol. 95, pp. 215-233, 2007.

[5] W. Zeng and M. Y. Chow, "Optimal Tradeoff Between Performance and Security in Networked Control Systems Based on Coevolutionary Algorithms," *IEEE Transactions on Industrial Electronics*, vol. 59, pp. 3016-3025, 2012.

[6] W. Zeng and M.-Y. Chow, "Modeling and Optimizing the Performance-Security Tradeoff on D-NCS Using the Coevolutionary Paradigm," *IEEE Transactions on Industrial Informatics*, vol. 9, pp. 394-402, 2013.

[7] W. Ren, B. R. W., and A. E. M., "A survey of consensus problems in multi-agent coordination," in *American Control Conference, 2005. Proceedings of the 2005*, 2005, pp. 1859-1864 vol. 3.

[8] F. Pasqualetti, A. Bicchi, and F. Bullo, "Consensus Computation in Unreliable Networks: A System Theoretic Approach," *IEEE Transactions on Automatic Control*, pp. 1-1, 2011.

[9] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, pp. 988-1001, 2003.

[10] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, pp. 382-401, 1982.

[11] D. Dolev, "The Byzantine generals strike again," *Journal of Algorithms*, vol. 3, pp. 14-30, 1982.

[12] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, pp. 398-461, 2002.

[13] S. Sundaram and C. N. Hadjicostis, "Distributed Function Calculation via Linear Iterative Strategies in the Presence of Malicious Agents," *IEEE Transactions on Automatic Control,*, vol. 56, pp. 1495-1508, 2011.

[14] A. Teixeira, H. Sandberg, and K. H. Johansson, "Networked control systems under cyber attacks with applications to power networks," in *American Control Conference (ACC), 2010*, 2010, pp. 3690-3696.

[15] W. Zeng, M.-Y. Chow, and P. Ning, "Secure distributed control in unreliable D-NCS," in *2012 IEEE International Symposium on Industrial Electronics (ISIE)*, 2012, pp. 1858-1863.

[16] W. Ren and R. W. Beard, *Distributed Consensus in Multi-vehicle Cooperative Control*: Springer, 2008.

[17] S. Sundaram, J. Chang, K. K. Venkatasubramanian, C. Enyioha, I. Lee, and G. J. Pappas, "Reputation-based networked control with data-corrupting channels," in *Proceedings of the 14th international conference on Hybrid systems: computation and control*, Chicago, IL, USA, 2011, pp. 291-300.

[18] R. Ismail and A. Josang, "The beta reputation system," in *Proceedings of the 15th Bled Conference on Electronic Commerce*, 2002, p. 41.

[19] R. Wei and R. W. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *Automatic Control, IEEE Transactions on*, vol. 50, pp. 655-661, 2005.

[20] M. Franceschelli, M. Egerstedt, and A. Giua, "Motion probes for fault detection and recovery in networked control systems," in *American Control Conference, 2008*, 2008, pp. 4358-4363.