ABSTRACT

ZENG, WENTE. Secure Distributed Control Methodologies with Built-in Defense in Distributed Networked Control Systems. (Under the direction of Dr. Mo-Yuen Chow.)

Distributed Networked Control Systems (D-NCS) are spatially distributed systems that integrate distributed sensors, actuators, and computing processors over a communication network for a vast amount of applications, such as electrical power systems and transportation systems. While most D-NCS have been safe in the past, they are increasingly more vulnerable to malicious cyber attacks and malwares with the rapid advancements and uses with networking, embedded systems, wireless communication technologies, and novel control strategies. In particular, more and more distributed control algorithms are being used in D-NCS because of their flexibility, robustness, computation, and communication features. These algorithms, however, increase the vulnerability of D-NCS to malicious cyber attacks. Thus, there is an urgent growing need to protect control algorithms from malicious cyber attacks in D-NCS.

This thesis considers the fundamental task of reaching an agreement (i.e., consensus) among a group of agents via secure distributed computations in D-NCS and discusses the problem of designing secure distributed control methodologies that are capable of performing secure distributed computations in the presence of misbehaving agents. First, we develop the mathematical models of D-NCS and misbehaving agents in the network, and we explore the vulnerabilities of conventional linear consensus algorithms. Second, we propose a reputation-based secure distributed control algorithm with built-in defense mechanism for leader-follower consensus network. It includes four phases (detection, mitigation, identification, and

update) into the control process in a distributed manner and is able to achieve an accurate consensus computation in the presence of misbehaving agents in D-NCS. Third, we extend the proposed algorithm to the leaderless consensus network by introducing and adding two recovery schemes (rollback recovery and excitation recovery) into the current secure distributed control framework to guarantee the accurate convergence of the well-behaving agents in D-NCS. At each phase, every agent only uses local and one-hop neighbors' information to identify and isolate the misbehaving agents, and even compensate their effect to the system. Fourth, we develop a trade-off model and corresponding quantitative metrics to address the performance and security trade-off problem of D-NCS. A paradigm of multi-agent trade-off optimization based on the Coevolutionary Genetic Algorithm (CGA) is proposed to optimize the trade-off between system real-time performance and security levels. Finally, in order to analyze the performance of our theoretical design on a real-world problem, we examine and validate the effectiveness of the proposed techniques in several illustrative case studies through both simulations and experiments.

Secure Distributed Control Methodologies with Built-in Defense
in Distributed Networked Control Systems

by
Wente Zeng

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Electrical Engineering

Raleigh, North Carolina

2013

APPROVED BY:

____
Huiyang Zhou

____
James R. Wilson

____
Mo-Yuen Chow
Chair of Advisory Committee

____
Huaiyu Dai

DEDICATION

To My Parents

Lisha Zeng and Ke Li

BIOGRAPHY

Wente Zeng was born and raised in Guangzhou (Canton), a southern city in China. After finishing his high school, he went to Shanghai for college study. He received his Bachelor of Science degree in automation in 2006, and his Master of Science degree in control science and engineering in 2009, both from Shanghai Jiao Tong University, Shanghai, China. He is currently a Ph.D candidate in Electrical Engineering in the Department of Electrical and Computer Engineering at North Carolina State University (NCSU).

Wente joined the Advanced Diagnosis, Automation and Control (ADAC) Laboratory at NCSU under the direction of Dr. Mo-Yuen Chow in fall 2009. His research interest includes networked control system, distributed control, Cyber-Physical System security and large-scale multi-objective optimization in Smart Grid and Intelligent Transportation System applications.

ACKNOWLEDGEMENTS

This dissertation would not have been completed without valuable guidance and help from many people and I am extremely fortunate to have got them all along the completion of my PhD studies.

My first and deepest thanks go to my advisor Dr. Mo-Yuen Chow. Dr. Chow has always been very patient, supportive, and encouraging for my research work, both from an academic and a human point of view. His guidance and enthusiasm have allowed me to discover the joy and gratification of scientific research. He also provided me many chances to get involved in the professional community and business training, which helps me grow up as a young researcher as well as an ambitious entrepreneur. I am truly grateful to him for making my graduate studies an unforgettable experience.

I am grateful to my committee members, Dr. Huiyang Zhou, Dr. James R. Wilson and Dr. Huaiyu Dai for our constructive interactions. Their questions and feedback in my oral exams shaped my research and helped me improve my work. I am also indebted to Dr. Peng Ning, who suggested me with the initial idea of my thesis topic. I am thankful to him for his revealing advice that made my research work novel and unique.

I would like to thank my dear friends in the ADAC Lab, who encouraged me to challenge compelling problems and helped me complete my research achievements. Thank you Ziang Zhang for your inspiring work on the incremental cost consensus algorithm in Smart Grid. Thank you Habiballah Rahimi Eichi for your insight and contribution in battery management systems. Thank you Unnati Ojha for our endless discussions about iSpace system and networked control systems. I have been very fortunate to work with you all, and good luck to your future career outside the university. My gratitude also goes to the other past and present lab members for numerous stimulating discussions and exciting group activities.

My thanks for my family will never be enough. I would like to thank my father Lisha Zeng and my mother Ke Li for their support, trust and comforting words in all these years.

Finally, thank you Frances, for your incredible patience, your sweet smile, your tireless perseverance, and for the endless love you give to my life.

TABLE OF CONTENTS

Content Page

TABLE OF CONTENTS vi

LIST OF TABLES xii

LIST OF FIGURES xiv

CHAPTER 1. INTRODUCTION 1

1.1 Importance of D-NCS Security 1

1.2 Enabling Technologies 2

1.2.1 Advancements in Distributed Control Methodologies 2

1.2.2 Advancements in Networking, Communication and Embedded System Technologies 3

1.2.3 Advancements in Intrusion Distributed Detection Technologies 3

1.3 Challenges and Opportunities of D-NCS Security 4

1.3.1 Traditional IT Security vs. D-NCS Security 4

1.3.2 Uniqueness of D-NCS Security 5

1.4 The Need for Novel Approaches and Paradigm Shift 6

1.5 Contributions of This Thesis 7

CHAPTER 2. BACKGROUND KNOWLEDGE 11

2.1 Definition of D-NCS 11

2.2 Definition of Consensus Network 13

2.2.1 Graph Theory Notations 14
2.2.2 First-Order Linear Consensus Algorithm 15
2.3 Present State of Knowledge in D-NCS Security 16
CHAPTER 3. D-NCS MODELING AND FORMULATION 20
3.1 D-NCS Model 20
3.2 Consensus Model of D-NCS 23
3.3 Misbehaving Agent Models 25
CHAPTER 4. SECURE DISTRIBUTED CONTROL ALGORITHM FOR LEADER-FOLLOWER CONSENSUS NETWORK 28
4.1 Algorithm Description 29
4.1.1 Detection Phase 30
4.1.2 Mitigation Phase 31
4.1.3 Identification Phase 32
4.1.4 Update Phase 32
4.2 Convergence and Robustness Analysis 34
4.2.1 Convergence Proof 35
4.2.2 Robustness Analysis 37
4.3 Simulation Case Studies 39
4.3.1 Simulation Testbed Description and Setup 39
4.3.2 Simulation Conditions 42
4.3.2.1 Case 1: Single Misbehaving Robot 43

4.3.2.2 Case 2: Multiple Misbehaving Robots 44
4.3.3 Simulation Results: Vulnerability Analysis 44
4.3.4 Simulation Results: Effectiveness Validation 47
4.3.5 Simulation Results: Robustness Analysis 50
4.4 Experiment Case Studies...51
4.4.1 Experimental Platform Setup and Configuration 51
4.4.2 Experiment Results 54
CHAPTER 5. RESILIENT DISTRIBUTED CONTROL ALGORITHM FOR LEADERLESS CONSENSUS NETWORK 57
5.1 Algorithm Description...57
5.1.1 Rollback Recovery 59
5.1.2 Excitation Recovery 59
5.2 Simulation Case Studies..61
5.2.1 Multi-Robot Formation Control: Leader-Follower Case 61
5.2.2 Multi-Robot Formation Control: Leaderless Case 64
5.2.3 Wireless Sensor Network: Faulty Sensor Case 66
5.2.4 Wireless Sensor Network: Malicious Sensor Case 68
CHAPTER 6. TRADE-OFF MODELING FOR PERFORMANCE AND SECURITY IN NCS 70
6.1 Problem Formulation...71
6.1.1 The Local DC Motor System 72

6.1.2 The Networked Controller 73

6.1.3 The Security Mechanism....74

6.1.4 The Communication Network....76

6.2 The Performance-Security Trade-off Model....76

6.2.1 The Performance Metric....76

6.2.2 The Security Metric....78

6.2.3 Performance and Security Requirements 79

6.2.4 The Trade-off Objective Function....80

6.3 Simulation Case Study: Networked DC Motor System....81

6.3.1 Testbed Description....81

6.3.2 Test Conditions 83

6.3.3 Simulation Results: Static Case....84

6.3.4 Simulation Results: Dynamic Case 86

CHAPTER 7. PERFORMANCE AND SECURITY TRADE-OFF OPTIMIZATION IN NETWORKED DC MOTOR SYSTEMS 89

7.1 Trade-off Model for Networked DC Motor system....90

7.1.1 The Performance Metric....90

7.1.2 The Security Metric....90

7.1.3 The Trade-off Objective Function....91

7.2 Optimizing Performance-Security Trade-off Based on CGA 93

7.2.1 Coevolutionary Genetic Algorithm....94

7.2.2 Performance-Security Trade-off Model based on CGA 96

7.2.3 Pareto Optimal Model based on CGA..99

7.3 Simulations and Experiments..101

7.3.1 Secured Networked DC Motor System Test-bed Setup........................102

7.3.2 Networked DC Motor System Simulator Setup...................................103

7.3.3 Simulation Results: Trade-off model 106

7.3.4 Simulation Results: Pareto Optimal Model..108

7.3.5 Simulation Results: Comparison 109

7.3.6 Experiment Results: 111

CHAPTER 8. PERFORMANCE AND SECURITY TRADE-OFF OPTIMIZATION IN D-NCS 114

8.1 iSpace System Model 116

8.1.1 UGV Dynamics 117

8.1.2 Path-Tracking Controller...117

8.1.3 Security Manager 119

8.1.4 Bandwidth Allocation 119

8.2 Trade-off Model Formulation 121

8.2.1 The Performance Metric...121

8.2.2 The Security Metric..121

8.2.3 The Trade-off Objective Function...123

8.3 Trade-off Optimization Using the CGA...124

8.3.1 Coevolutionary Genetic Algorithm 124
8.3.2 Performance-Security Trade-off Model Based on CGA 125
8.4 Simulations 128
8.4.1 Test-Bed Description...128
8.4.2 Simulation Conditions 130
8.4.3 Simulation Results: Different CGA Parameters....................................132
8.4.4 Simulation Results: Different Time Delays 134
CHAPTER 9. CONCLUSION AND FUTURE WORK 136
9.1 Summary 136
9.2 Directions for Future Research 137
9.2.1 Secure Distributed Control in the Presence of Colluding Misbehaving Agents 138
9.2.2 Secure Asynchronous Distributed Control Algorithms 139
9.2.3 Applications in Smart Grid Secure Distributed Operations 140
References 141

LIST OF TABLES

Table Page

Table 4-1 Simulation Parameters 43

Table 4-2 Initial Positions (X-coordinates) of Robots 43

Table 5-1 Initial Positions (X-coordinates) of the Robots 61

Table 5-2 Readings of the Temperature Sensors 66

Table 6-1 Induced Time-Delay with respect to Security Algorithms 82

Table 6-2 The Parameters of The DC Motor 83

Table 6-3 Paired T-Test Comparing the Mean Performance and Security Levels of Different Encryption Algorithms 86

Table 6-4 Percentage Reduction in Performance Level using Different Encryption Algorithms 86

Table 6-5 The Security Mechanisms Configuration 87

Table 7-1 The Pseudo-code of CGA 95

Table 7-2 The Definitions of Performance and Security Agents in The Trade-off Model 98

Table 7-3 The Definitions of Performance and Security Agents in Pareto Optimal Model. 100

Table 7-4 The Key Variables and Parameters of CGA 105

Table 7-5 Test Cases' CGA Parameters 106

Table 7-6 Nash Equilibrium Results of Different Cases 106

Table 7-7 Paired T-Test Comparing The Mean Optimal Performance and Security Values of Different Optimization Algorithms 110

Table 7-8 Percentage Increase in Optimal Minimum Performance and Security Values using Other Optimization Algorithms 110

Table 7-9 Averaged Results of Contrast Experiments 111

Table 7-10 The Result of Step Responses for Different Settings 111

Table 8-1 The Definition of Performance and Security Agents in the Trade-Off Model..... 126

Table 8-2 The Operational Bounds of Agents 132

Table 8-3 The CGA Parameters of the Four Test Cases 133

Table 8-4 Nash Equilibrium Results of the CGA Process 134

Table 8-5 Percentage Improvement in Performance and Security When Using CGA Optimal Bandwidth Allocation 135

LIST OF FIGURES

Figure Page

Figure 2.1 The conceptual model structure of the D-NCS 11

Figure 2.2 A simple framework for a consensus network 13

Figure 3.1 The framework of the NCS consensus model 22

Figure 3.2 Misbehaving agent models in D-NCS 26

Figure 4.1 Flowchart of the secure distributed control algorithm for the leader-follower consensus network 29

Figure 4.2 Control block diagram of the secure distributed control algorithm 33

Figure 4.3 Overall structure of the iSpace Simulator 38

Figure 4.4 Differential drive UGV reference frame and parameters 40

Figure 4.5 The Simulink block diagram of the iSpace Simulator.. 42

Figure 4.6 The communication topology of the multi-robot system 43

Figure 4.7 Multi-robot system in the presence of one misbehaving robot 44

Figure 4.8 Multi-robot system in the presence of two non-colluding misbehaving robots 44

Figure 4.9 Multi-robot formation control result using linear consensus algorithm in the presence of one faulty robot.. 45

Figure 4.10 Multi-robot formation control result using linear consensus algorithm in the presence of one malicious robot 45

Figure 4.11 Multi-robot formation control result using linear consensus algorithm in the presence of two faulty robots 46

Figure 4.12 Multi-robot formation control result using linear consensus algorithm in the presence of two malicious robots 46

Figure 4.13 Multi-robot formation control result using the proposed secure distributed control algorithm in the presence of one faulty robot 48

Figure 4.14 Multi-robot formation control result using the proposed secure distributed control algorithm in the presence of one malicious robot 48

Figure 4.15 Multi-robot formation control result using the proposed secure distributed control algorithm in the presence of two faulty robots 49

Figure 4.16 Multi-robot formation control result using the proposed secure distributed control algorithm in the presence of two malicious robots 49

Figure 4.17 Multi-robot formation control result using the proposed secure distributed control algorithm in the presence of one faulty robot in the critical link. 51

Figure 4.18 System structure of the iSpace system experimental platform 53

Figure 4.19 Multi-UGV rendezvous experiment results using the proposed secure distributed control algorithm in the presence of one faulty UGV 54

Figure 4.20 Multi-UGV rendezvous experiment results using the proposed secure distributed control algorithm in the presence of one malicious UGV 55

Figure 4.21 Multi-UGV rendezvous experiment results using the proposed secure distributed control algorithm in the presence of two faulty UGVs 55

Figure 4.22 Multi-UGV rendezvous experiment results using the proposed secure distributed control algorithm in the presence of two malicious UGVs.. 56

Figure 5.1 Flowchart of the secure distributed control algorithm for the leaderless consensus network 58

Figure 5.2 The communication topology of the multi-robot system in leader-follower case. 62

Figure 5.3 The leader-follower multi-robot formation control results in the presence of a faulty robot using conventional linear consensus algorithm.. 63

Figure 5.4 The leader-follower multi-robot formation control results in the presence of a faulty robot using the proposed secure distributed control algorithm 63

Figure 5.5 The leaderless multi-robot formation control results using the proposed secure distributed control algorithm in the presence of a faulty robot without excitation recovery.. 65

Figure 5.6 The leaderless multi-robot formation control results using the proposed secure distributed control algorithm in the presence of a faulty robot with excitation recovery....... 65

Figure 5.7 The average consensus results using the proposed secure distributed control algorithm in the presence of two faulty sensors without rollback recovery 67

Figure 5.8 The average consensus results using the proposed secure distributed control algorithm in the presence of two faulty sensors with rollback recovery 67

Figure 5.9 The average consensus results using the proposed secure distributed control algorithm in the presence of two malicious sensors without rollback recovery 68

Figure 5.10 The average consensus results using the proposed secure distributed control algorithm in the presence of two malicious sensors with rollback recovery 69

Figure 6.1 The structure of the secured networked DC motor system 72

Figure 6.2 State flow of the security mechanism based on discrete events 75

Figure 6.3 Simulink based test-bed for the secured networked DC motor system................. 82

Figure 6.4 The system performance and security levels using different encryption algorithms 85

Figure 6.5 The comparison between the required security level and actual security level of the secured networked DC motor system 87

Figure 6.6 The trade-off between the system performance and security of the secured networked DC motor system 88

Figure 7.1 The conceptually admissible operating region of the model................................ 92

Figure 7.2 The performance-security trade-off model based on CGA 97

Figure 7.3 Framework of the CGA optimization process.. 98

Figure 7.4 Secured Networked DC Motor System Test-bed 103

Figure 7.5 Networked DC Motor System Simulator based on Matlab Simulink 104

Figure 7.6 Variations of performance fitness values in CGA process................................. 107

Figure 7.7 Variations of security fitness values in CGA process 108

Figure 7.8 Pareto front of the Pareto optimal model based on CGA 109

Figure 7.9 The step responses of the test-bed for different settings 112

Figure 8.1 The performance-security trade-off model of D-NCS based on the CGA 126

Figure 8.2 Framework of the CGA optimization process.. 128

Figure 8.3 Test paths for the simulations... 131

Figure 8.4 Network delay histogram for delay generator RTT settings of 60ms, 200ms, 400ms, and 600ms 131

Figure 8.5 The path tracking results of case #2 133

CHAPTER 1. INTRODUCTION

1.1 Importance of D-NCS Security

Distributed Networked Control Systems (D-NCS) can be spatially distributed systems for which the communication among distributed sensors, distributed actuators, and distributed controllers is supported by a shared communication network [1]. As a type of Cyber-Physical Systems (CPS), these systems integrate computing and communication capabilities with monitoring and control of entities in the physical world. The widespread growth of wireless communication and embedded systems technologies have been creating a wide variety of applications for D-NCS – in areas such as defense systems, medical devices, autonomous vehicles, and smart structures of national critical infrastructures (e.g., the electrical power system/Smart Grid, transportation systems).

D-NCS have been at the core of national critical infrastructures and industrial control systems for many decades. While most D-NCS have been safe in the past, there have been a few confirmed cases of cyber-based attacks [2]. The recent presence of the infamous industrial control system malwares "Stuxnet" and “Flame” have further brought about significant attentions on making industrial control systems safe from such malicious cyber attacks [3]. 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 [4]. For instance, the electrical power grid is evolving into the Smart Grid currently. The goal of the Smart Grid is to improve efficiency and availability of power by adding more communication and control capabilities. Adding a layer of control and communication to the Smart Grid in places where it does not now exist are certain to introduce increased vulnerabilities in the power grid to cyber attacks. Many D-NCS applications are time-sensitive, data-sensitive, and safety-critical; their failure may cause irreparable harm to the physical system being controlled and to the people who depend on it. In this sense, D-NCS perform vital functions in national critical infrastructures, such as nuclear reactor power plants, electric power distribution, transportation systems, and water and gas distribution [5, 6]. They also play a role in military applications, such as missile defense systems, radar and weapon control systems, and military robotic systems [7]. The potential consequences of compromising a D-NCS can be devastating to public health and safety, national security, and the economy. Compromised D-NCS can, and have, led to extensive cascading power outages, dangerous toxic chemical releases, and explosions [2, 6]. It is, therefore, important to implement D-NCS with embedded security controls that make reliable, safe, and flexible performance possible.

1.2 Enabling Technologies

1.2.1 Advancements in Distributed Control Methodologies

A variety of novel distributed control algorithms have been gaining popularity in D-NCS because of their flexibility, communication features, and computational performance compared to conventional centralized control strategies. More recently, researchers from various disciplines have shown a tremendous surge of interest in D-NCS challenges with close ties to distributed control. This includes subjects such as consensus [8, 9], algebraic connectivity of complex networks [10, 11], graph theory [12-14], flocks and swarms [15, 16], coordinated tasks [17, 18], asynchronous distributed algorithms and gossip-based algorithms [19, 20], multi-robot systems formation control [21-23], cooperative control [24, 25], and sensor networks [26-28].

1.2.2 Advancements in Networking, Communication and Embedded System Technologies

In the last few decades, there have been significant innovations and advances of technologies in areas such as networking technology (e.g., ad hoc network) [29], embedded systems (e.g., FPGA), communications technologies (e.g., the Internet, wireless communications), which give us the luxury to have much more freedom in how we operate the D-NCS.

1.2.3 Advancements in Intrusion Distributed Detection Technologies

Distributed intrusion detections in NCS have also received attention recently. Existing literature on distributed intrusion detections can be broadly divided into three categories: (1) Parallel (fusion) architectures, where all sensors transmit their measurements or local decisions to a fusion node, which subsequently makes the final decision [30]; (2) Consensus-based detections, where no fusion node is required and sensors communicate with single-hop neighbors only over a generic network [31]; and (3) Distributed detections, where, based on consensus-based detections, the sensing and communication phases of the nodes occur in the same time step [32, 33].

These practical methods, tools, and technologies based on advances in the fields of communications, computation, control, and security are enabling D-NCS and other infrastructures to become complex and sophisticated.

1.3 Challenges and Opportunities of D-NCS Security

1.3.1 Traditional IT Security vs. D-NCS Security

Generally, there is a significant difference between the security philosophies of traditional IT and D-NCS. The purpose of IT security is to protect the data residing in the servers from attack. The purpose of D-NCS security is to protect the ability of the facility to safely and securely operate, regardless of what may befall the rest of the network. Therefore, the D-NCS employs the best practices associated with the CIA triad model in the reverse order: Availability, Integrity, and Confidentiality [34]. Extra emphasis is placed on availability and message integrity. Their detailed interpretations in D-NCS are:

- Availability: Maintain the operational goals by tolerating and surviving malicious attacks to the information collected by the sensor networks, the commands given by the controllers, the physical actions taken by the actuators and the control algorithms.
- Integrity: Maintain the operational goals by preventing or detecting deception attacks in the information and control command sent and received by the sensors, the controllers, and the actuators, or the control algorithm in the system.

- Confidentiality: Prevent an adversary from inferring with the state of the control system by eavesdropping on the communication channels between the sensors and the controller, and between the controller and the actuator.

1.3.2 Uniqueness of D-NCS Security

Compared to the Conventional IT security, the new characteristics that may be present in D-NCS security are [35].:

- Input and possible feedback from the physical environment
- Distributed management and control
- Uncertainly regarding readings, status, and trust.
- Real-time performance requirements
- Wide-distribution geographically, with components in locations that lack physical security.
- Multi-scale and systems of systems control characteristics.

Among all these characteristics present in D-NCS security, distributed management and control is of most importance. Most of D-NCS such as the power grid [36], involve management by multiple parties. They are heterogeneous interconnected systems with some highly coupled local modes and some loosely coupled inter-area modes with stringent resource constraints. Each agent may have a different dynamics. Control algorithms operating within the D-NCS need to be simple, distributed, robust against disturbances, and efficient in resource utilization, and need to provide proper real-time responses. Thus, the legacy centralized control/management schemes may encounter severe challenges when applied to the D-NCS because most of D-NCS are usually highly dynamic and geographically distributed. In order to effectively control and manage the D-NCS, more and more distributed control algorithms are being used in D-NCS because of their flexibility, robustness, and communication and computation requirements. These algorithms, however, increase the vulnerability of D-NCS to malicious cyber attacks, as well. Because of the lack of a centralized entity that may monitor the activity of the nodes in the network, distributed control strategies are prone to attacks and component failures. Thus, it is of increasing importance to guarantee that computations are secure and trustworthy even in the presence of misbehaving devices. Besides, most of the current effort for protecting D-NCS, e.g., Supervisory Control And Data Acquisition (SCADA) systems, has been accomplished with prevention mechanisms (e.g., cryptography and authentication), which are limited to platform and communication security [37]. Thus, there is an urgent concern to protect the control algorithms from malicious cyber attacks.

1.4 The Need for Novel Approaches and Paradigm Shift

One of the most frequent tasks to be accomplished by autonomous agents in D-NCS is to agree upon a certain performance measure, such as the workload on a network of parallel computers, the clock speed for wireless sensor networks, and the velocity or formation pattern for a group of autonomous vehicles [38, 39]. 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) [40]. Let us take the linear consensus algorithm as an example in which each node updates its state as a weighted combination of its own value and the values received from its neighbors at each time instant [41]. The choice of topologies and weights influences the convergence speed toward the consensus value [42]. In the linear consensus algorithm, all the nodes are assumed to cooperate and follow the protocol exactly. However, due to the lack of a centralized controller that may monitor the activities of all the nodes in the network, distributed control algorithms are prone to attacks and component failures. For instance, if one node in the consensus network is compromised by a malicious cyber attacker, the consensus is not guaranteed to be reached. Thus, it is important to guarantee secure computation in the face of failures and intrusions in the distributed control algorithms.

1.5 Contributions of This Thesis

In this research, we consider the problem of reaching an agreement (i.e., consensus) among a group of agents via secure distributed control in D-NCS. We explore the vulnerabilities of distributed control algorithms and design secure distributed control algorithms in the presence of misbehaving agents in D-NCS. We further examine and extend the proposed algorithms into different network topologies: leader-follower and leaderless consensus networks. In summary, the main contribution of this thesis is developing the secure distributed control and management algorithms and analytic frameworks for D-NCS.

In order to analyze the performance of our theoretical design on a real-world problem, we also analyze and validate the proposed techniques in different case studies through simulations and experiments.

The main contributions of each chapter are as follows:

Chapter 2 introduces some preliminary definitions, the notations, and a comprehensive literature survey of related works about D-NCS security. This chapter provides the background knowledge for the research work discussed in the subsequent chapters.

Chapter 3 discusses the mathematical problem formulation and the modeling of malicious cyber attacks in D-NCS. The D-NCS model and the corresponding consensus model are both introduced. The misbehaving agents are modeled by formulating the malicious attacks as exogenous inputs that compromise the behavior of agents in D-NCS. Two types of misbehaving agents are introduced, namely, faulty agent and malicious agent, depending on the nature of the cyber attacks they suffer.

Chapter 4 is concerned with the problem of designing a secure distributed control methodology that is capable of performing a secure distributed control computation in a D-NCS in the presence of misbehaving agents. A secure distributed control algorithm is first proposed for the leader-follower consensus network. It embeds four security phases into the control process in a distributed manner: (a) Detection Phase: detect the neighbors' misbehaviors relying only on each agent's local observations; (b) Mitigation Phase: adjust the consensus computation weights according to the neighbors' reputation values; (c) Identification Phase: identify and isolate the compromised agent; and (d) Update Phase:

update the reference state using the adjusted consensus computation weights to ensure the convergence of well-behaving agents. All phases of the proposed method are distributed. That is, at each step of the detection, mitigation, identification, and update, each agent only uses local and one-hop neighbors' information. The performance and constraints of the proposed method is illustrated through convergence proof, robustness analysis and simulation results. An experiment-based hardware-in-the-loop testbed for multi-UGV formation control is also introduced and rendezvous tasks are used to illustrate the effectiveness of the proposed method.

Chapter 5 continues the work on developing a secure distributed control methodology that is able to perform secure distributed control computations in the presence of misbehaving agents in D-NCS. In this chapter, the proposed secure distributed control algorithm is extended to the leaderless consensus network by introducing and adding two recovery schemes (rollback recovery and excitation recovery) as a Recovery Phase into the current secure distributed control framework to guarantee accurate convergence of the well-behaving agents in D-NCS. The effectiveness of the proposed method is demonstrated through case studies in multi-robot formation control and wireless sensor networks.

Chapter 6 addresses the trade-off between NCS security and its real-time performance and use a secured networked DC motor system for illustration. This chapter presents a trade-off model for system dynamic performance and system security. This model can be used to adapt security configurations to provide sufficient protection and satisfy real-time dynamic performance requirements of the NCS simultaneously. The construction of this model includes the development of a set of metrics to quantitatively measure the performance and security levels of NCS and the development of a trade-off objective function incorporating performance and security. A Simulink based test-bed implemented to control the speed of the DC motor is used to illustrate the effectiveness of this model.

Chapter 7 presents a framework of performance-security trade-off optimization based on Coevolutionary Genetic Algorithm (CGA) for the networked DC motor system based on the trade-off model introduced in chapter 6. Experiments show that CGA is highly efficient in finding the Nash equilibrium for the trade-off model on NCS. The framework presented in this chapter is also demonstrated to be an effective approach for performance-security trade-off analysis and optimization on NCS.

Chapter 8 investigates the trade-off between system real-time performance and security in D-NCS and uses the Intelligent Space (iSpace) for illustration. A multi-agent trade-off model for D-NCS' dynamic performance and security is presented. This model can be used to allocate system resources to provide sufficient protection and to satisfy the D-NCS's real-time dynamic performance requirements simultaneously. A paradigm of multi-agent performance-security trade-off optimization based on the CGA for D-NCS is proposed as well. A Simulink-based test-bed is implemented to illustrate the effectiveness of this paradigm. The results of the simulation show that the CGA can efficiently find the optimal values in a multi-agent performance-security trade-off model for D-NCS.

Chapter 9 concludes the thesis and discusses some directions for future research in the area of secure distributed control and secure D-NCS applications.

CHAPTER 2. BACKGROUND KNOWLEDGE

2.1 Definition of D-NCS

D-NCS are control systems wherein the control loops are closed through a real-time network, which integrates computing and communication capabilities with monitoring and control of entities in the physical world. These systems are usually composed by a set of networked agents, including: distributed sensors, distributed actuators, distributed controllers and communication network, as shown in Figure 2.1.

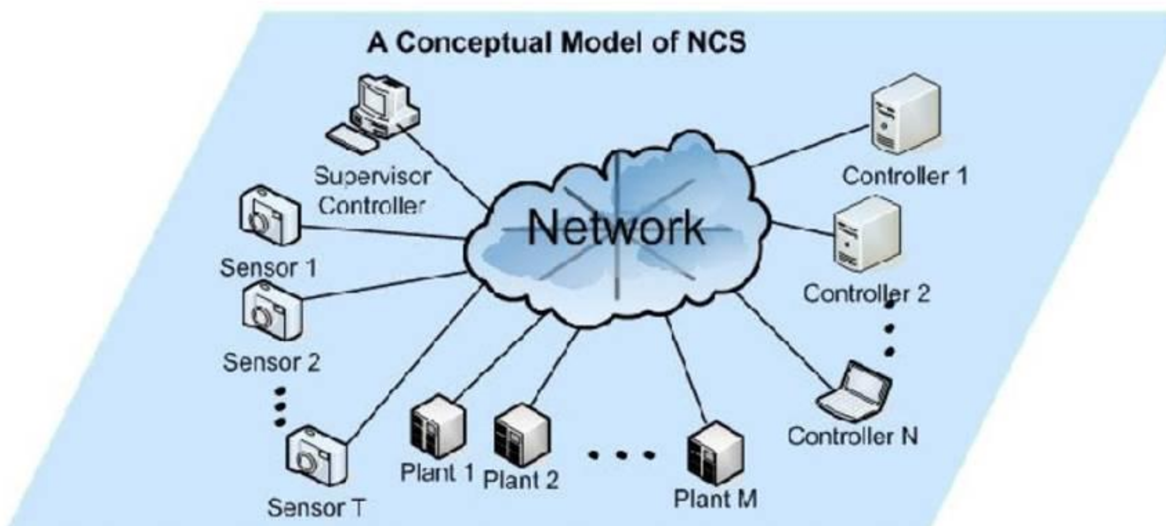

Figure 2.1 The conceptual model structure of the D-NCS

The root of control systems can be traced back to 1868 when dynamics analysis of the centrifugal governor was conducted by the famous physicist J.C. Maxwell [43]. The most significant achievement in conventional control systems occurred when the Wright Brothers made their first successful test flight in 1903. The next significant achievement was the fly-by-wire flight control system that was designed to eliminate the complexity; fragility and weight of the mechanical circuit of the hydro-mechanical flight control systems using an electrical circuit. The simplest and earliest configuration of an analog fly-by-wire flight control systems was first fitted to the Avro Vulcan in the 1950s. This can be called as the first form of analog Networked Control Systems (NCS). Digital computers became powerful tools in control system design and microprocessors added a new dimension to the capability of control systems. A modified NASA F8C Crusader was the first digital fly-by-wire aircraft, in 1972. The next step in evolution was the Distributed Control System (DCS) that was introduced in 1975. Both Honeywell and Japanese electrical engineering firm Yokogawa introduced their own independently produced DCSs at around the same time, with the TDC 2000 and CENTUM systems, respectively. As the expanding needs of industrial applications pushed the limit of point-to-point control, it became obvious that NCS was the solution to achieve remote control operations. Research in tele-operation was initiated with the concern for safety and convenience in hazardous environments; such as space projects and nuclear reactor power plants, and was made feasible only after further development of NCS.

Later, with the advent of networking technologies, easy and cheap access to the Internet (previously known as ARPANET), proved to be a boon. Thomas L. Friedman emphasizes the effects of Internet in human activities in his book *The World is Flat*. Further development and research in NCS was boosted by the tremendous increase in the deployments of wireless systems in the last few years. Today, NCS is moving into Distributed NCS (D-NCS) [44], which are multi-disciplinary efforts whose aim is to produce a network structure and components that are capable of integrating distributed sensors, distributed actuators, and distributed control algorithms over a communication network in a manner suitable for real-time applications.

2.2 Definition of Consensus Network

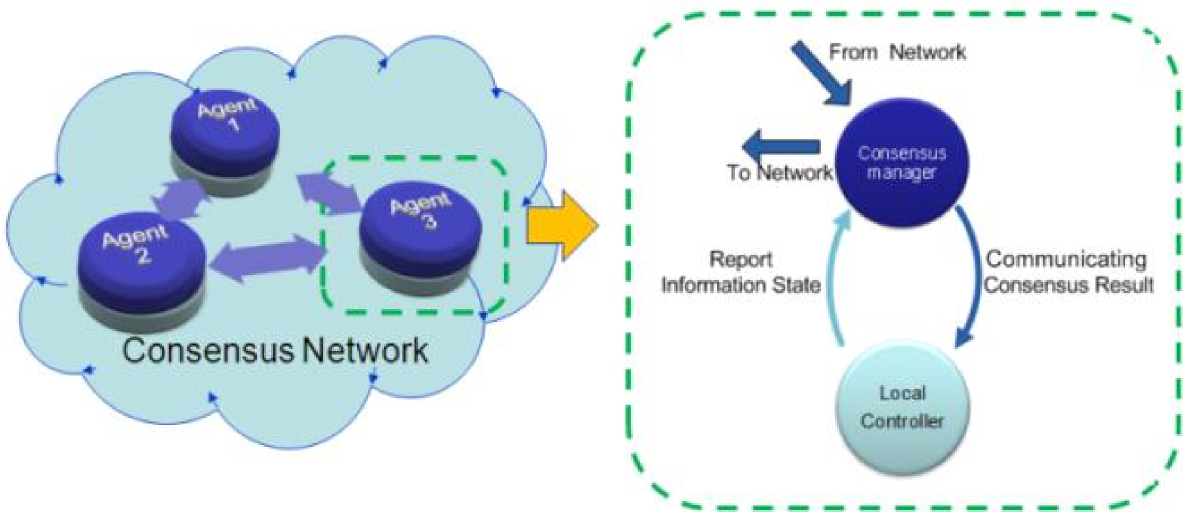

Figure 2.2 A simple framework for a consensus network

Figure 2.2 shows an example framework for a consensus network, where there are three independent physical systems (agents). In this network, each agent has two parts: a local controller and a consensus manager. The local controller constantly reports its information state to the consensus manager, which negotiates with neighboring agents, among others, via the communication network. The consensus manager then calculates a consensus result and passes it back to the local controller, which adjusts its own action based on the consensus result. Then, the local controller reports its new state to the consensus manager. This process continues as the system operates, which is an iterative process.

2.2.1 Graph Theory Notations

A graph G is used to model the network topology of the system. The graph G is a pair of sets (V, E), where V is a finite non-empty set of elements, called *vertices* or *nodes*, and E is a set of unordered pairs of distinct vertices, called *edges*. A simple graph is an unweighted, undirected graph that contains no graph loops or no multiple edges. A graph is connected if there is a path between any distinct pair of nodes. A directed tree is a digraph, where every node has exactly one parent except for one node, called the *root*, which has no parent, and where the root has a directed path to every other node. The directed spanning tree of a digraph is a directed tree formed by graph edges that connect all of the graph's nodes. A graph contains a directed spanning tree if one exists as a subset of the graph.

The adjacency matrix A of a finite graph G on n vertices is the $n \times n$ matrix where the off-diagonal entry a_{ij} is the number of edges from vertex i to vertex j. In the case of a finite simple graph, the adjacency matrix is a (0, 1) matrix with zeros on its diagonal. If the graph is undirected, the adjacency matrix is symmetric. Let matrix L=[l_{ij}] be:

$$l_{ii} = \sum_{i \neq j} a_{ij} \text{ , for on-diagonal elements;} \tag{1}$$

$$l_{ij} = -a_{ij} \text{, for off-diagonal elements.} \tag{2}$$

For an undirected graph, L is called the *Laplacian matrix*, which has the property that it is symmetric, positive, semi-definite. The eigenvalue of the L matrix contains a lot of information about network performance [40] (e.g., the second smallest eigenvalue of Laplacian λ_2 is called the *algebraic connectivity* of the graph). The algebraic connectivity of the network topology is a measure of the convergence speed of the consensus algorithms [45].

2.2.2 First-Order Linear Consensus Algorithm

Let χ_i denote the state valuable of node i. The state value of a node might represent physical quantities including coordinates, velocity, etc. The nodes of a network reach a consensus if and only if $\chi_i = \chi_j$ for all i, j [46]. Assuming that each agent has a first-order dynamic:

$$\dot{\chi}_i = \chi_i, i = 1, \ldots, n, \tag{3}$$

a continuous-time consensus algorithm is given in [45], [46] as:

$$\dot{\chi}_i = -\sum_{j=1}^{n} a_{ij} (\chi_i - \chi_j), i = 1, \ldots, n, \tag{4}$$

where a_{ij} is the *(i, j)* entry of the adjacency matrix A. The consensus algorithm can also be written in matrix form as:

$$\dot{\chi} = -L_n \chi, \tag{5}$$

where L_n is the $n \times n$ graph Laplacian matrix.

When the measurement information is instantaneously available, a continuous-time model can be used to describe the consensus network dynamics. When information takes a fixed time T to travel between nodes, we need to model the consensus network dynamics as a discrete-time dynamic system [23] to facilitate analysis. A discrete-time consensus algorithm is described by:

$$\chi_i[k+1] = \sum_{j=1}^{n} d_{ij} \chi_j[k], i = 1,\ldots,n, \tag{6}$$

where k is the discrete-time index; d_{ij} is the *(i, j)* entry of the row-stochastic matrix D, which can be defined as follows:

$$d_{ij} = l_{ij} \Big/ \sum_{h=1}^{n} |l_{ih}|, i, j = 1,\ldots,n. \tag{7}$$

Since the data packet in a communications network always arrives discretely, the discrete consensus algorithm has been selected for further investigation and development.

2.3 Present State of Knowledge in D-NCS Security

Since D-NCS without security protection is vulnerable to various security attacks, developing security mechanisms for D-NCS is an urgent fundamental research problem to be addressed. Dzung et al. [47] first gave an overview of IT security issues in industrial automation based on open communication systems and also explained various countermeasures. Cardenas et al. [34] identified and defined the problem of secure control in D-NCS, and proposed a set of challenges that need to be addressed to improve the D-NCS's ability to survive. Kim et al. [48] also gave an overview of the challenges and ongoing efforts in the field of cyber-physical security with specific emphasis on a Smart Grid infrastructure.

Many NCS security issues have been widely studied in the related fields and many solutions have been proposed [49-51]. In the computer science community, the security problems of consensus protocols has been extensively studied in distributed computing as the famous "Byzantine Generals Problem" [52]. That problem deals with the characterization of the resilience of distributed systems to malicious (Byzantine) nodes. It is 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: 1) is less than 1/2 of the network connectivity, and 2) is less than 1/3 of the total number of nodes in the network [53]. This result has been regarded as a fundamental constraint of a distributed consensus protocol to sustain malicious nodes in the network and many relevant Byzantine Fault Tolerance (BFT) algorithms have been proposed in the context of distributed computing [54] [55]. However, the continuity and local dynamics of the individual agent have not been considered and discussed in those approaches, which are salient in control system applications. Furthermore, Byzantine nodes are allowed to change their state to any value and are allowed to send different messages to different nodes in distributed computing, while in most control scenarios, all the nodes including the malicious nodes are assumed to send the same information to all their neighbors since their state information is normally obtained through sensing or broadcast communication.

Researches on the security issues of distributed control algorithms are still in their infancy. In the literature several results can be found, most of which deal with security issues of the linear consensus algorithm from a control theoretical perspective. The problem of characterizing the resilience properties of linear consensus strategies has been partially addressed in recent works [39, 56, 57], where it has been shown that despite the limited ability of the misbehaving agents, the resilience to external attack is still limited by the connectivity of the network. Pasqualetti et al. [56] first introduced the problem of detecting and identifying misbehaving agents in a linear consensus network, and a solution was proposed for the single faulty agent case. Sundaram and Hadjicostis [57] provided one policy that k malicious agents can follow to prevent some of the nodes on a 2k-connected network from computing the desired function of the initial state or from reaching an agreement. Pasqualetti et al. [39] extended and improved the results along these routes by characterizing a complete set of policies that make a set of k malicious agents undetectable or unidentifiable and by providing the resilience bounds in the case of faulty agents. Teixeira et al. [58] proposed a distributed scheme to detect and isolate the cyber attacks on the communication network of D-NCS using observers and discussed how to reduce the number of observer nodes while maintaining the coverage of the entire network. Acemoglu et al. [59] addressed the security problem from the perspective of social networks, providing a model to investigate the tension between information aggregation and the spread of misinformation in large societies.

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. Moreover, the intrusion detection schemes and the control algorithms are always separated. There is no feedback to the control parameters when the compromised nodes are detected or identified. Instead, we want to develop secure distributed control algorithms that have been set up with an eye toward quality and security design in the first place which is much more likely to remain secure and less vulnerable to hacking, intrusion or malicious control.

CHAPTER 3. D-NCS MODELING AND FORMULATION

3.1 D-NCS Model

Typically, D-NCS are large-scale complex network systems with time-driven and event-driven features. We view the D-NCS with three interactive layers:

- Top layer: cyber layer (communications network) that links the information passing among the physical devices;
- Middle layer: control/management middleware interacting with the communications and physical layer;
- Bottom layer: physical layer that contains the actual physical devices.

We use the agent concept and control theory to model the overall problem under consideration. Each node (e.g., an autonomous vehicle, a power generator) in the D-NCS will be modeled as an agent with its own dynamics, described as:

$$x_{i,k+1} = f_i(x_{i,k}, u_{i,k}) + f_i^d(\cdot), \tag{8}$$

where $X_i = [x_{i,1} \quad x_{i,2} \quad \cdots \quad x_{i,n_i}]^T$ contains the state variables (e.g., velocity, position); $u_i = [u_{i,1} \quad u_{i,2} \quad \cdots \quad u_{i,m_i}]^T$ is the control (e.g., the PWM signal) of the i^{th} agent with dimensions n_i and m_i, respectively; k is the time index; and f_i^d is the disturbance (e.g., wind level for a period) on the i^{th} agent. The entire D-NCS state variable $\boldsymbol{x}$ is the collection of all the N agent state variables:

$$X = [X_1 \quad X_2 \quad \cdots \quad X_N]^T, \tag{9}$$

and the overall D-NCS dynamics is described by :

$$f = [f_1 \quad f_2 \quad \cdots \quad f_N]^T . \tag{10}$$

Without loss of generality, we assume the measurement on the i^{th} agent is of the form:

$$Y_i = C_i X_i \ , \tag{11}$$

where $\boldsymbol{C_i}$ is the measurement matrix of agent i and the entire measurement of the D-NCS is:

$$Y = [Y_1 \quad Y_2 \quad \cdots \quad Y_N]^T , \tag{12}$$

We use connection matrix W_c to describe the topology of the communications network. For example, the updating matrix D in the discrete-time consensus algorithm can be considered as W_c matrices.

The D-NCS constraints, such as equality constraints (e.g., power balance), inequality constraints, (e.g., stability), and communications/sensing constraints (e.g., bandwidth limitations) can be incorporated into the model.

Based on the local measurements (those on i^{th} agent and its connected neighbor's measurements $W_c(i,:)\ Y$, we can develop distributed control algorithms relying only on local information and collaborating with each other:

$$u_i = h_i\ (W_c\ (i,:)\ Y(t+\tau_i)), \tag{13}$$

where t is time and τ_i is the network-in-the-loop time delay to signify the time delay in measurements over the network, and h_i is the distributed algorithm under investigation. The vector u_i represents the interactions among agents and their influences on each other.

This three-layer model can provide an effective mathematical formulation and framework for subsequent tasks on finding the analytical control design for the desired secure distributed control algorithms. We can construct a variety of representative D-NCS configurations and operating conditions based on this model.

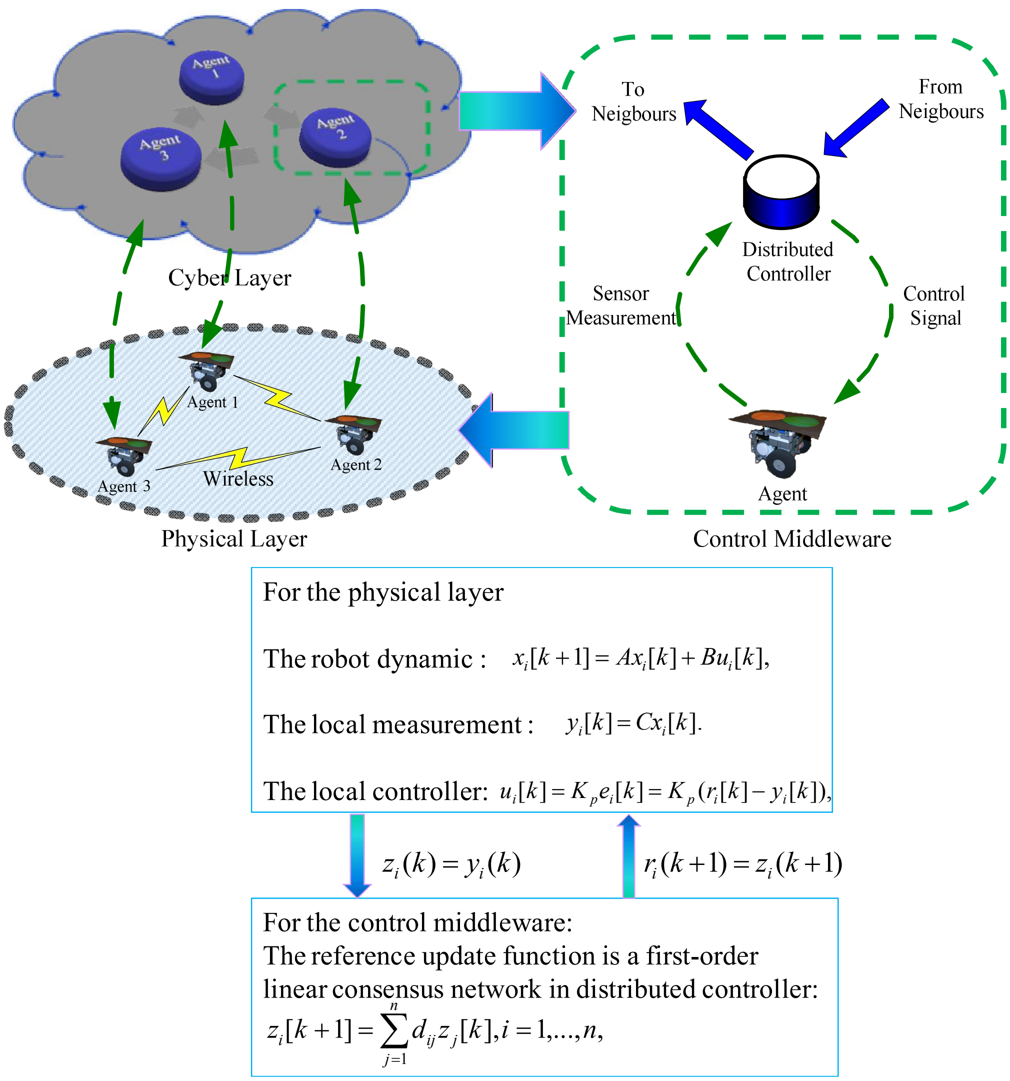

Figure 3.1 The framework of the NCS consensus model

3.2 Consensus Model of D-NCS

Reaching consensus, which refers to reaching agreement among all the autonomous agents, is one of the most frequent control tasks to be accomplished in the D-NCS. This fundamental problem forms the foundation of distributed computing, and admits many important applications in the areas of communications, control, and networking where information sharing and operation coordination are desirable. In this section, the consensus model of D-NCS is formulated.

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

$$x_i[k+1] = A_i x_i[k] + B_i u_i[k], \tag{14}$$

where x_i and u_i are the agent states and controls and i is the index for the agents in the network [23].

Each agent receives the following measurements

$$y_i[k] = C_i x_i[k]. \tag{15}$$

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

$$u_i[k] = K_i e_i[k] = K_i (r_i[k] - y_i[k]), \tag{16}$$

where r_i is the reference state and K_i 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 a wireless communications network, of which the topology is denoted by a directed graph G. In this local neighbor-to-neighbor information exchange process, the first-order linear consensus algorithm is implemented in the consensus manager of each agent, where, 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 3.1.

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

$$z_i[k] = y_i[k], \tag{17}$$

$$z_i[k+1] = \sum_{j=1}^{n} d_{ij} z_j[k], i = 1,...,n, \tag{18}$$

$$r_i[k+1] = z_i[k+1], \tag{19}$$

where d_{ij}, the consensus computation weight, is the (i, j) entry of the row-stochastic matrix D, which is derived from the adjacency matrix of network topology (graph G). z_i is the information transmitted in the communication layer.

By following the update rule described in (18), all the agents will converge to a common state asymptotically, and thus the consensus task is achieved 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 *algebraic connectivity* of the graph) [40].

3.3 Misbehaving Agent Models

The misbehaving agent in the D-NCS is modeled as one whose distributed controller (consensus manager) is suffering a unexpected fault or under a malicious cyber attack. We model the misbehaving agent in the system as:

$$z_i[k+1] = \sum_{j=1}^{n} d_{ij} z_j[k] + B_{i,M} u_{i,M}[k], \tag{20}$$

where $u_{i,M}$ is the anomaly factor that is modeled as an exogenous input to the consensus manager, as shown in Figure 3.2.

By the definition above, when the agent is behaving well, the corresponding exogenous input $u_{i,M}$ is constantly equal to zero. When the agent is misbehaving, it is allowed to update its reference state in an arbitrary way by choosing different exogenous input $u_{i,M}$. Therefore, depending on the nature of various misbehaviors, the misbehaving agent can be classified into two categories.

One type of misbehaving agent is the faulty agent, which simply stops to update its reference state and behaves like a stopping failure (e.g., out of power), thus the exogenous input $u_{i,M}$ can be modeled as:

$$u_{i,M}[k] = c_i - \sum_{j=1}^{n} d_{ij} z_j[k], \tag{21}$$

where c_i is an arbitrary constant value and $c_i \in R$.

Another type of misbehaving agent is the malicious agent, which means the agent is under malicious cyber attacks (e.g., false data injection attack, deception attack) and thus it may behave in an arbitrary manner. Thus, the exogenous input $u_{i,M}$ can be equal to any arbitrary value $c_i[k]$ that the attacker wants:

$$u_{i,M}[k] = c_i[k]. \tag{22}$$

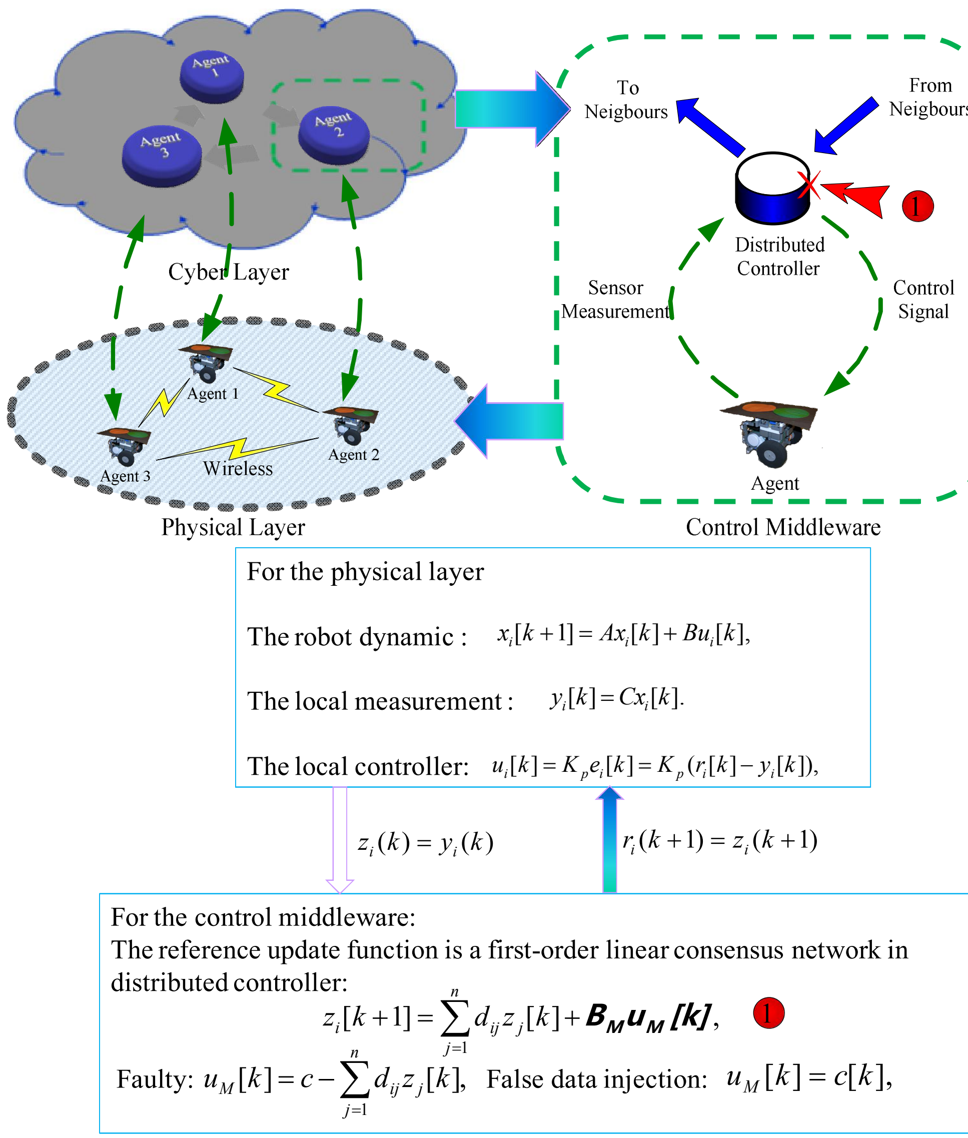

Figure 3.2 Misbehaving agent models in D-NCS

Note that the choice of keeping the matrix D fixed and leaving the exogenous inputs u_M unspecified also models the case in which the malicious attacks modify some entries of the adjacency matrix of the network topology, as in the case of unreliable communication channels.

CHAPTER 4. SECURE DISTRIBUTED CONTROL ALGORITHM FOR LEADER-FOLLOWER CONSENSUS NETWORK

Because of the absence of a centralized controller that may monitor the activity of all the agents in the system, the D-NCS is prone to cyber attacks and component failures. Thus, it is important to guarantee a secure distributed control computation in the presence of misbehaving agents in D-NCS. Unlike the conventional methods in which the misbehaving agents are identified by a supervisory monitor, a reputation-base secure distributed control algorithm which embeds the security mechanism internally is proposed in this section to achieve security in D-NCS in a fully distributed fashion.

The secure distributed control algorithm embedded with built-in defenses we propose here has two unique properties: (a) It is a distributed method in which all agents detect their neighbors' misbehaviors by relying only on their local observations through Neighborhood Monitor; (b) It embeds the security mechanism into the consensus computation process by adjusting the control parameters according to the neighbors' reputation values, and identifying the compromised agents relying on its Local Reputation Manager. As a result, the proposed secure distributed control methodology can detect and isolate the compromised agent via each agent's local information in a distributed manner, while ensuring the convergence of well-behaving agents.

4.1 Algorithm Description

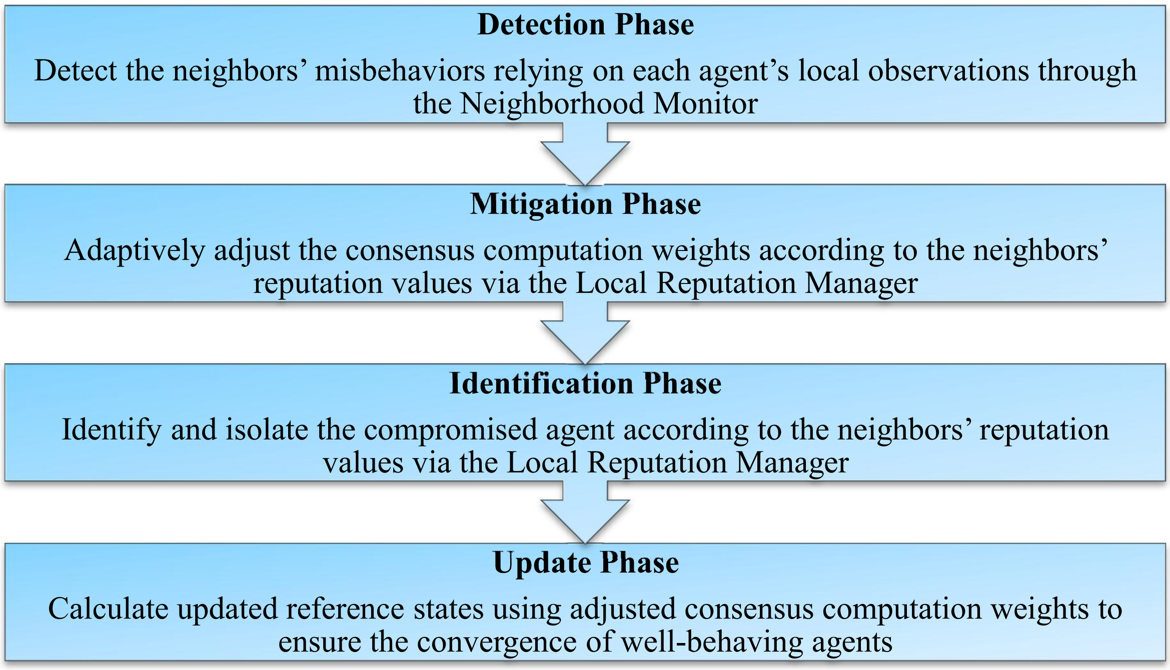

Figure 4.1 Flowchart of the secure distributed control algorithm for the leader-follower consensus network

The proposed secure distributed control algorithm for leader-follower consensus networks, as shown in Figure 4.1, embeds the following four phases into each iteration of the consensus computation (i.e., reference state update) process: (a) Detect the neighbors' misbehaviors relying only on each agent's local observations through the Neighborhood Monitor; (b) Adjust the consensus computation weights according to the neighbors' reputation values via the Local Reputation Manager; (c) Identify and isolate the misbehaving agent; and (d) 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:

4.1.1 Detection Phase

Definition (Neighborhood Monitor): A Neighborhood Monitor is an embedded monitor for an agent to observe the behaviors of its neighbors.

In this phase, with the assumption that listening to a neighbor's transmission is possible because of the characteristic of wireless communications, each agent constructs a Neighborhood Monitor to observe the 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) [60]. It redundantly calculates and stores its neighbor agent *j*'s reference state r_j and compares it with the state value z_j received from the neighbor agent *j* in time step *k*. If $r_j[k] - z_j[k] \leq \gamma(k,\delta)$, the neighbor is normal; if $r_j[k] - z_j[k] > \gamma(k,\delta)$, the neighbor has a potential anomaly.

$$G_{ij}[k] = \begin{cases} G_{ij}[k-1]+1, & r_j[k]-z_j[k] \leq \gamma(k,\delta) \\ G_{ij}[k-1], & r_j[k]-z_j[k] > \gamma(k,\delta) \end{cases}, \tag{23}$$

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.

4.1.2 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 [61, 62]. 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 [62]. The expected value of the *beta* distribution forms the reputation and is given by the ratio of the number of correct agent states received to the number of total agent states received:

$$rep_{ij}[k] = \frac{\eta G_{ij}[k] + 1}{\eta k + 2}, \tag{24}$$

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

4.1.3 Identification Phase

If the reputation value falls below a certain level (i.e., anomaly threshold rep_{th}), the corresponding neighbor agent is identified as a misbehaving agent and will be isolated. In that case, all the information from the misbehaving neighbor agent will be rejected. Nevertheless, a timeout mechanism may be used to allow the misbehaving agent to rejoin the network in case it might have been falsely accused in the past or if it behaves normally again. Actually, 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 anomaly 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 the reputation coefficient η .

4.1.4 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:

$$d_{ij}[k] = rep_{ij}[k] \Big/ \sum_{h=1}^{n} rep_{ih}[k]. \tag{25}$$

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

$$z_i[k+1] = \sum_{j=1}^{n} d_{ij}[k] z_j[k], i = 1, \ldots, n. \tag{26}$$

By implementing this method, we build a feedback loop from the output to the input of the distributed controller (consensus manager), as shown in Figure 4.2.

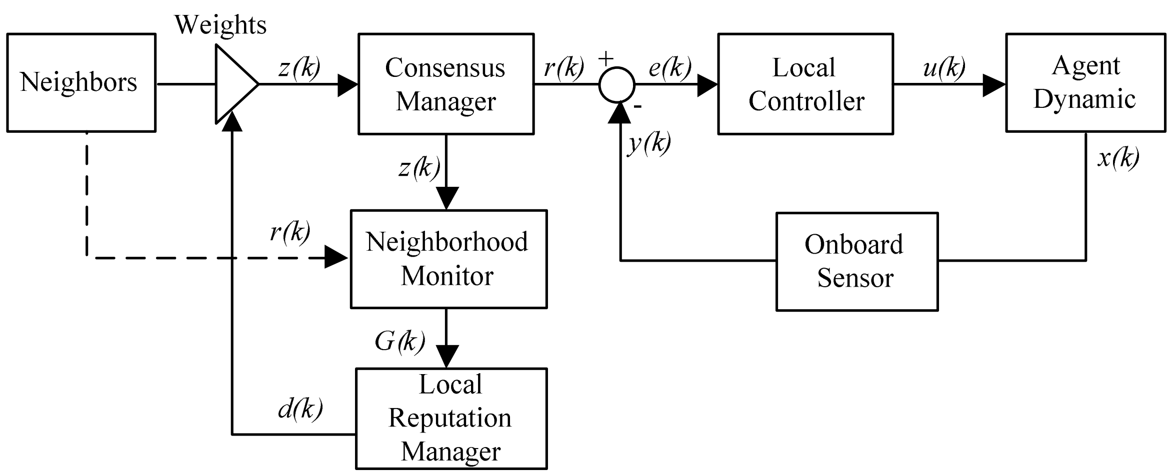

Figure 4.2 Control block diagram of the secure distributed control algorithm

In this secure distributed control algorithm, 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 misbehaving agent spreads. If a neighbor agent is identified as a misbehaving 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 misbehaving agent and converge to a steady state.

The proposed secure distributed control algorithm outperforms the conventional security mechanisms for consensus computation for the following reasons:

a) It is resilient to the single point of failure since no fusion node is required as a centralized intrusion detection scheme.

b) It relies only on each agent's own observations to avoid the new vulnerability of false accusations from second-hand information exchanges.

c) It embeds the security mechanism as a feedback loop inside the consensus computation, making the distributed control algorithm itself resilient and robust to the presence of misbehaving agents.

4.2 Convergence and Robustness Analysis

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

4.2.1 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 misbehaving), 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:

Firstly, we need the following lemmas from [41] and [63] to derive our proof.

Lemma 1:

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

$$D[k-1]D[k-2]\cdots D[2]D[1]D[0] \to 1c^T. \tag{27}$$

As $k \to \infty$, where **1** denote 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 $\{G_1, G_2, \cdots, G_m\}$ has a spanning tree, then the matrix product $D_m D_{m-1} \cdots D_2 D_1$ 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 algorithm, at each time-step $k \in N$, each node communicates with its neighbors and updates its local value using equations (25) and (26), where $d_{ij}[k]$ is the weight assigned to node j's value by node i at time-step k.

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

where V_i is the set of neighboring nodes of the i^{th} node, 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 $\alpha = \frac{rep_{th}}{V_i}$.

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

Also, assume 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 the 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 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.

4.2.2 Robustness Analysis

If the misbehaving agent is detected and isolated, it may disconnect the network into two parts (in some certain network topologies), where there is no hope for any consensus algorithm to be working. Thus before removing the misbehaving agent from the network, we need to check the network topology, more specifically, the connectivity of the network to avoid disconnection in the network. For example, if we identify one misbehaving agent in the network and we do not know the entire topology of the network, we need to make sure the network is at least 2-connected before isolating that agent.

To summarize, the proposed algorithm works with 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 due to the isolation of some critical nodes being misbehaving. Actually, the resilient algorithms using isolation techniques are always worst-case bounded by the connectivity of the network. If the connectivity of the network is m, the maximum number of the malicious nodes these algorithms can tolerate is m-1 in the worst case.

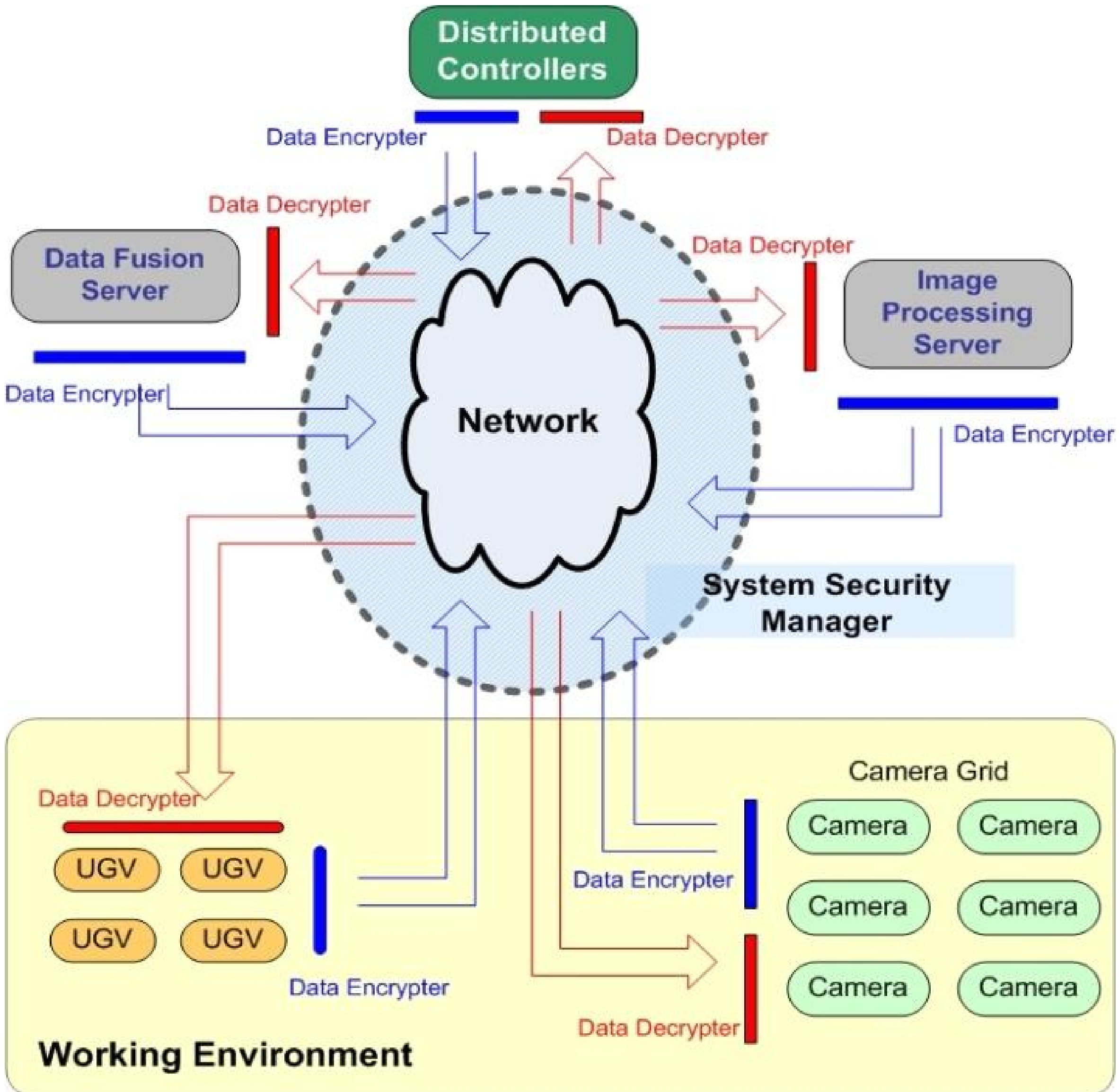

Figure 4.3 Overall structure of the iSpace Simulator

4.3 Simulation Case Studies

A Simulink-based simulation testbed, iSpace Simulator, is developed to evaluate the effectiveness of the proposed secure distributed control methodology.

4.3.1 Simulation Testbed Description and Setup

*i*Space Simulator has been implemented to perform basic research and education on time-sensitive and secure D-NCS applications. It is a network-based multi-robot system simulator including a network of mobile robots [64]. It has several major components: distributed sensors and actuators, hybrid hierarchical and distributed controllers, and the communication network. The system structure is shown in Figure 4.3.

In this testbed, mobile robots are represented by unmanned ground vehicles (UGVs). Each UGV has a differential drive with two driving wheels and one caster wheel, as in Figure 4.4. The dynamics of the UGV can be described using the kinematic model:

$$\begin{bmatrix} v_{ref} \\ \omega_{ref} \end{bmatrix} = \begin{bmatrix} \frac{\rho}{2} & \frac{\rho}{2} \\ \frac{\rho}{W} & -\frac{\rho}{W} \end{bmatrix} \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix}, \tag{28}$$

where $\boldsymbol{v}_{ref}$ is the UGV's linear velocity, $\boldsymbol{\omega}_{ref}$ is its angular velocity, ρ is the radius of the drive wheels, and W is the distance between the drive wheels. ω_r and ω_l represent the angular velocities of the right and left drive wheels, respectively.

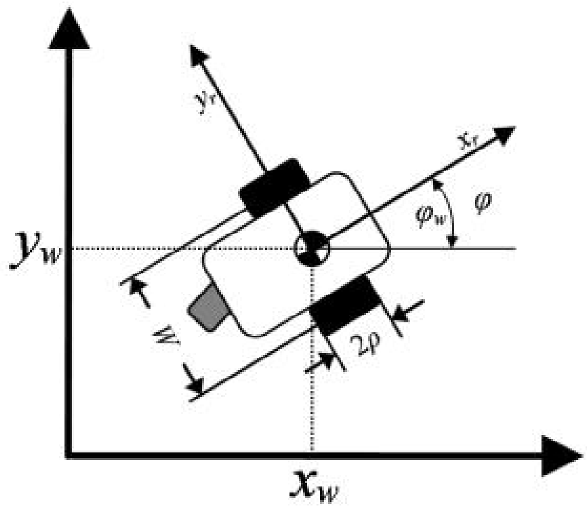

Figure 4.4 Differential drive UGV reference frame and parameters

The UGV steers by driving the wheels at different speeds determined by solving (28) where $\boldsymbol{v}_{ref}$ and $\boldsymbol{\omega}_{ref}$ are given in the input reference command $\boldsymbol{u}=[\boldsymbol{v}_{ref}\ \ \boldsymbol{\omega}_{ref}]^T$:

$$\begin{bmatrix} \omega_{R,ref} \\ \omega_{L,ref} \end{bmatrix} = \begin{bmatrix} \frac{1}{\rho} & \frac{W}{2\rho} \\ \frac{1}{\rho} & -\frac{W}{2\rho} \end{bmatrix} \begin{bmatrix} v_{ref} \\ \omega_{ref} \end{bmatrix}, \tag{29}$$

Using (29), the input reference command can be utilized to set reference speeds for the individual wheels. A Proportional-Integral (PI) controller, defined by (30), is used to achieve the reference speed on each wheel,

$$u(t) = K_P e(t) + K_I \int_0^t e(\xi) d\xi , \tag{30}$$

where K_P is the proportional gain, K_I is the integral gain, and $e(t)$ is the difference between the actual wheel speeds and the reference wheel speeds as shown in (31),

$$e(t) = \begin{bmatrix} \omega_{R,ref} - \omega_R \\ \omega_{L,ref} - \omega_L \end{bmatrix}, \tag{31}$$

The two control parameters in (30) are used to control the three UGV states defined by (32),

$$\alpha(t) = \begin{bmatrix} x_w(t) & y_w(t) & \varphi(t) \end{bmatrix}, \tag{32}$$

where $x_w(t)$ and $y_w(t)$ are the UGV coordinates in the world frame and $\varphi(t)$ is the UGV heading expressed as the angle between the UGV and the positive x axis of the world frame at time t.

This testbed lets us design repeatable experiments under a wide range of network environments, system conditions, and network settings. The Simulink block diagram of the testbed is shown in Figure 4.5.

This testbed can lead a team of autonomous robots using only local interactions and communications to accomplish different types of formation control tasks. In this simulation, we refer to a leader-follower structure, whereby a leader robot is assumed to have access to global information, and to lead the remaining follower robots through local consensus interactions.

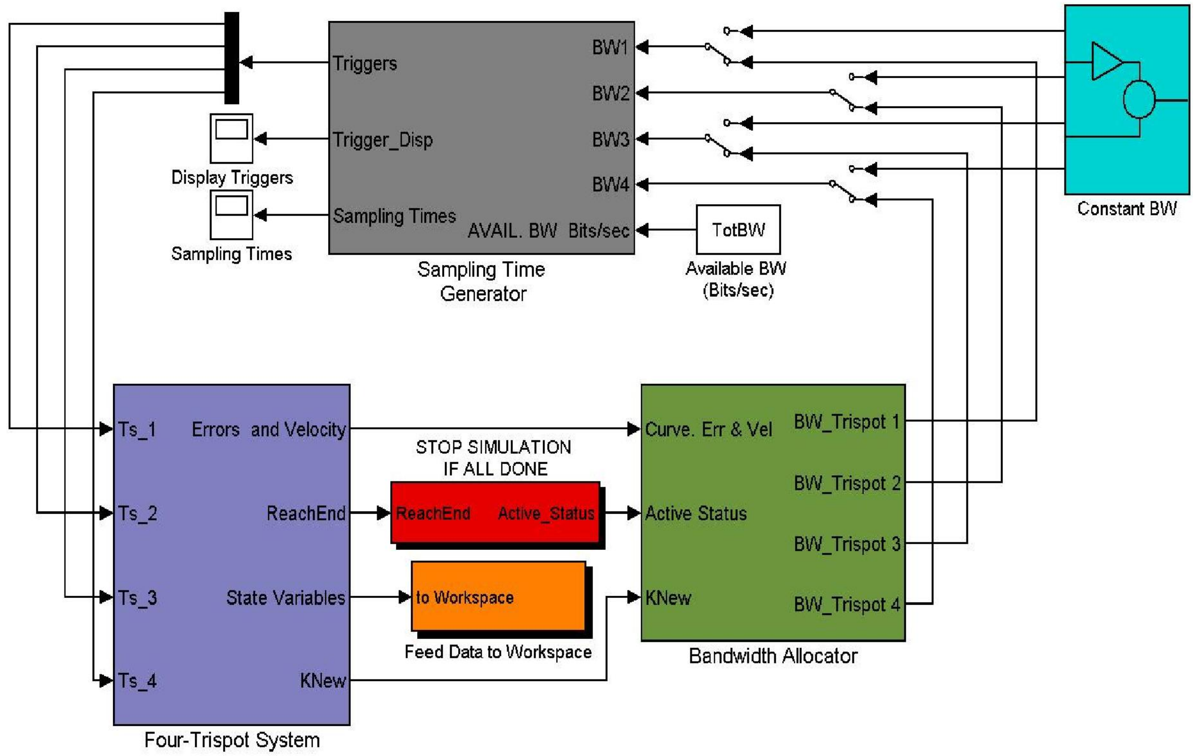

Figure 4.5 The Simulink block diagram of the iSpace Simulator

4.3.2 Simulation Conditions

To analyze the performance of the proposed secure distributed control methodology, a representative formation control task is designed—let all the robots stay in a line in parallel (reach the same *x*-coordinate: $x = 10$). Six robots are used in this task. Their communications topology is shown in Figure 4.6. Robot 1 is the leader while the other robots are the followers.

All the parameters of the simulation are listed in Table 4-1. The initial conditions *x(0)* of all the robots are shown in Table 4-2.

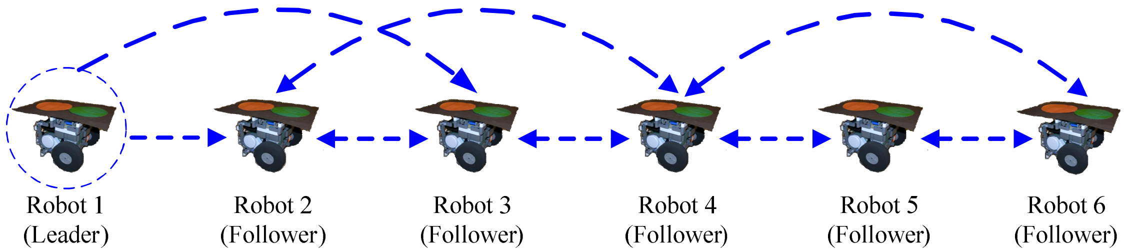

Figure 4.6 The communication topology of the multi-robot system

Table 4-1 Simulation Parameters

K_P	10	***rep_M***	0.3
c	5	***c(k)***	k+5
Time Length	40 seconds	***Time Step Size***	0.01 seconds

Table 4-2 Initial Positions (X-coordinates) of Robots

Robot 1	***Robot 2***	***Robot 3***	***Robot 4***	***Robot 5***	***Robot 6***
0	-5	5	-3	3	10

4.3.2.1 Case 1: Single Misbehaving Robot

In this simple scenario, let us assume Robot 5 is the misbehaving robot under malicious cyber attack, as shown in Figure 4.7.

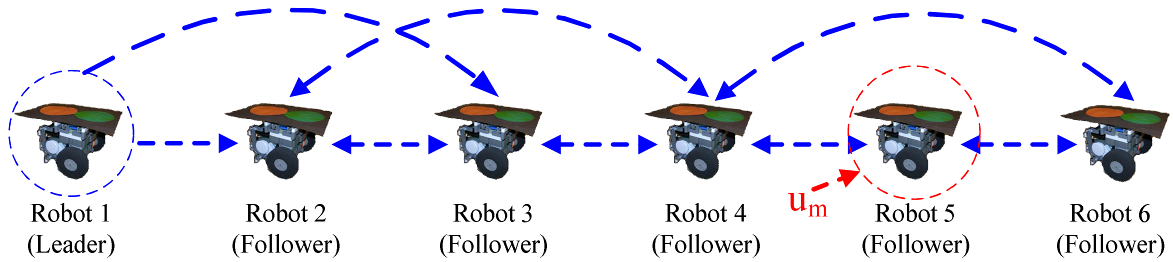

Figure 4.7 Multi-robot system in the presence of one misbehaving robot

4.3.2.2 Case 2: Multiple Misbehaving Robots

In this scenario, let us assume Robot 2 and Robot 5 both are the misbehaving robots under malicious cyber attacks, as shown in Figure 4.8.

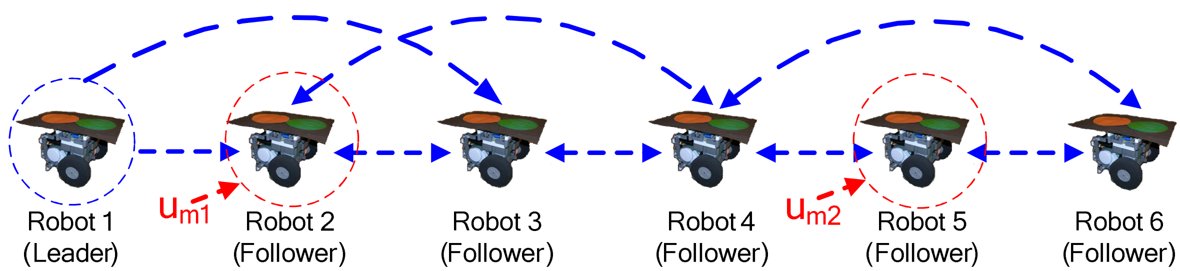

Figure 4.8 Multi-robot system in the presence of two non-colluding misbehaving robots

4.3.3 Simulation Results: Vulnerability Analysis

At first, we use the conventional linear consensus algorithm to perform the consensus computation (update the reference x-coordinates).

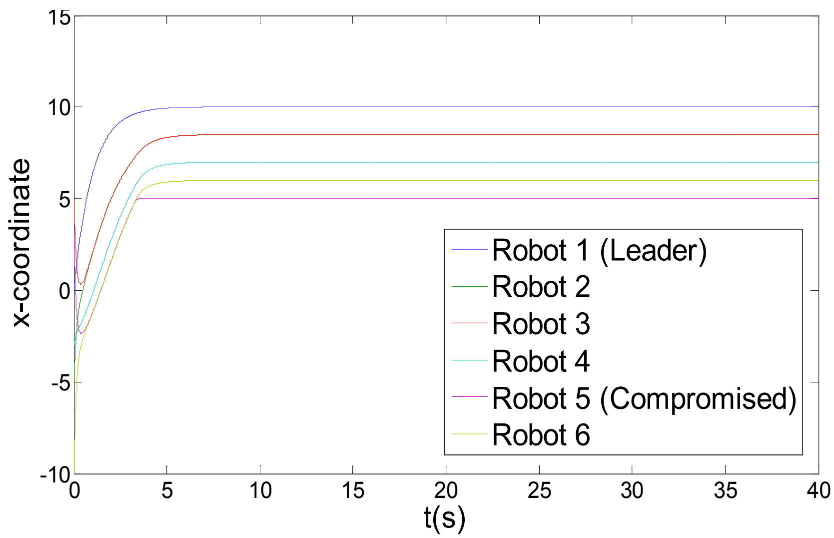

Figure 4.9 Multi-robot formation control result using linear consensus algorithm in the presence of one faulty robot

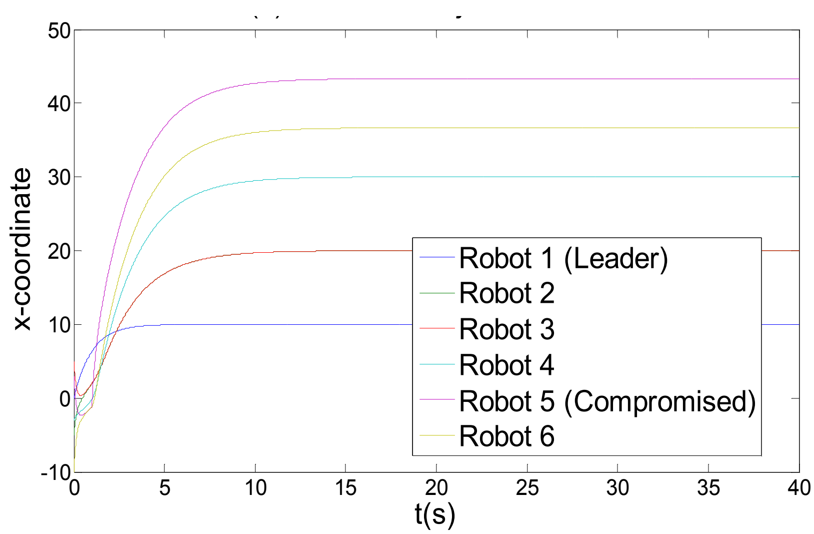

Figure 4.10 Multi-robot formation control result using linear consensus algorithm in the presence of one malicious robot

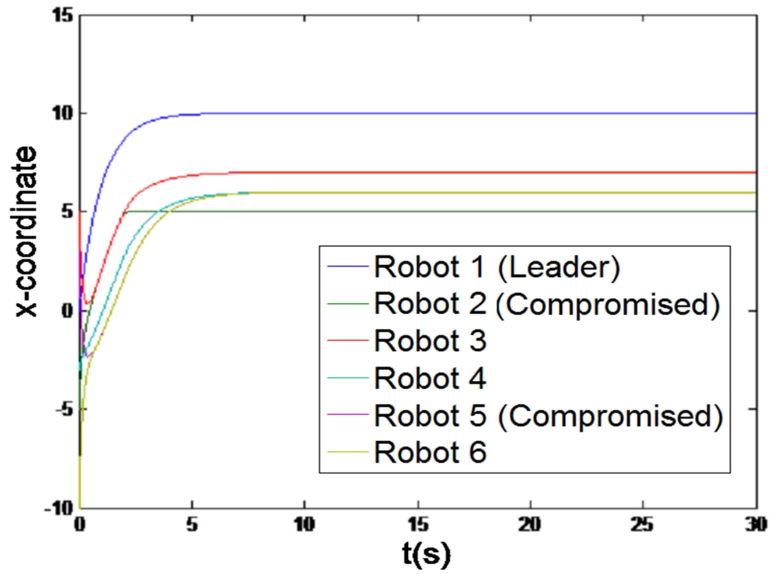

Figure 4.11 Multi-robot formation control result using linear consensus algorithm in the presence of two faulty robots

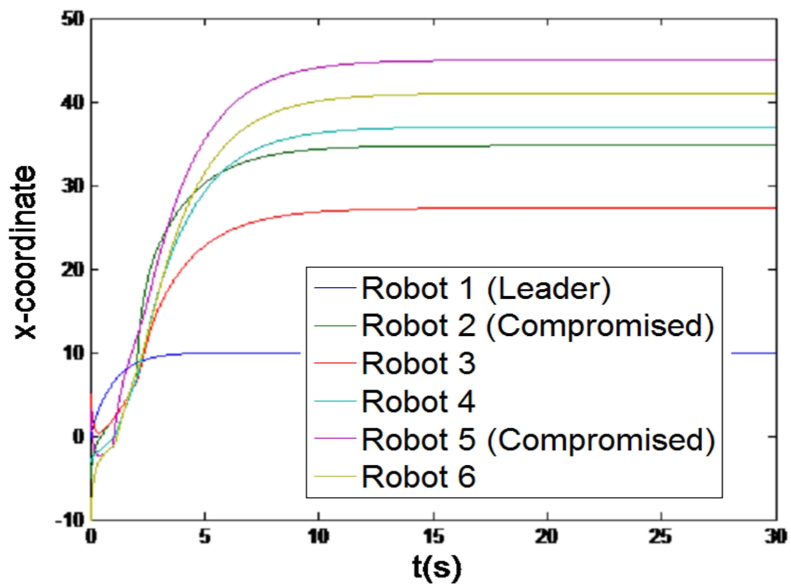

Figure 4.12 Multi-robot formation control result using linear consensus algorithm in the presence of two malicious robots

Figures 4.9 and 4.10 show the simulation results of the multi-robot formation control with a compromised robot under fault attack and false data injection attack in Case 1, respectively. As we can see from the figures, only the leader robot reached the desired destination while all the other follower robots stopped at different positions. The formation control task failed under both attacks. We can conclude that because of the absence of security mechanisms, the linear consensus algorithm cannot converge in the presence of a misbehaving robot.

Similar to Case 1, Figures 4.11 and 4.12 show the simulation results of the multi-robot formation control with two compromised robots under fault attack and false data injection attack in Case 2, respectively. As we can see, only the leader robot reached the desired destination while all the other follower robots stopped at different positions. Thus, the formation control task failed in both cases. We can conclude that because of the absence of security mechanisms, the linear consensus algorithm cannot converge in the presence of multiple misbehaving robots, as well.

4.3.4 Simulation Results: Effectiveness Validation

Let us carry out the same task again using the proposed secure distributed control methodology to perform the consensus computation.

As the simulation results show in Figure 4.13 and Figure 4.14, in Case 1, the misbehaving robot is gradually being isolated and the rest of the normal robots achieve the consensus successfully under two types of malicious attacks (faulty and false data injection attack). In Figure 4.15 and Figure 4.16, even with the presence of two compromised robots in Case 2, all the well-behaving robots can still converge at the correct destination asymptotically in both attack scenarios.

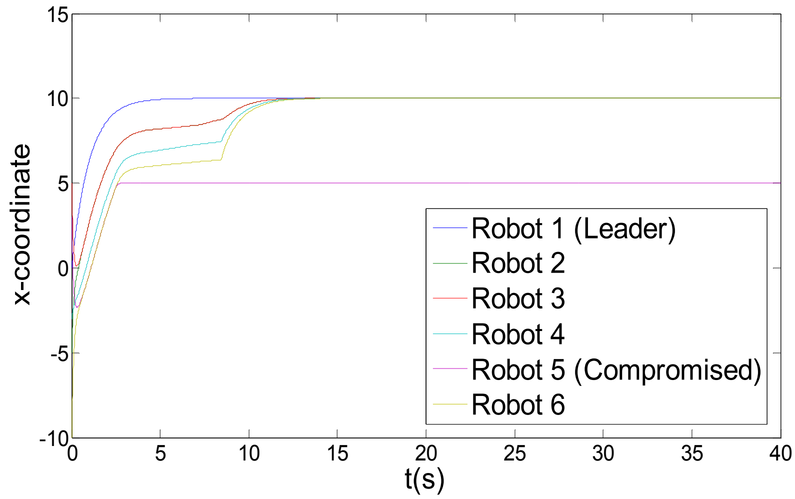

Figure 4.13 Multi-robot formation control result using the proposed secure distributed control algorithm in the presence of one faulty robot

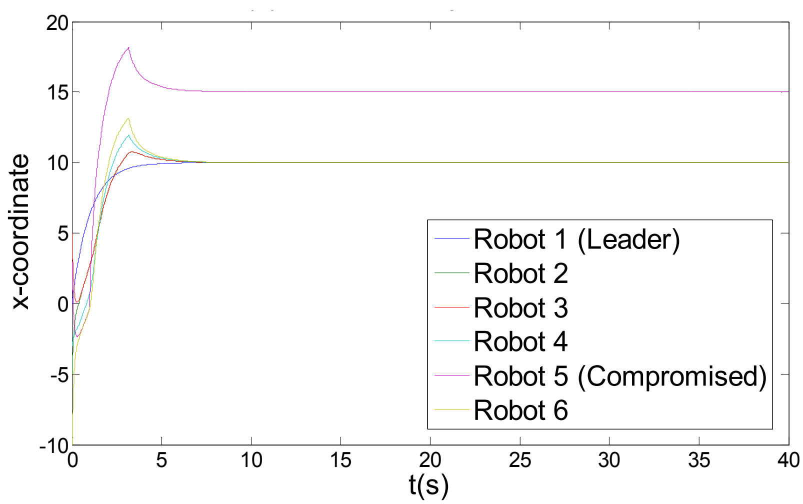

Figure 4.14 Multi-robot formation control result using the proposed secure distributed control algorithm in the presence of one malicious robot

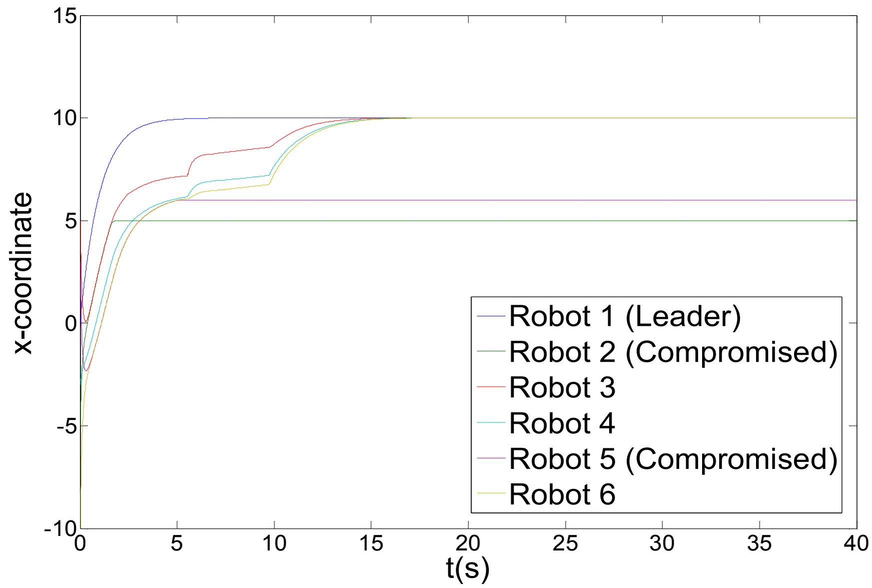

Figure 4.15 Multi-robot formation control result using the proposed secure distributed control algorithm in the presence of two faulty robots

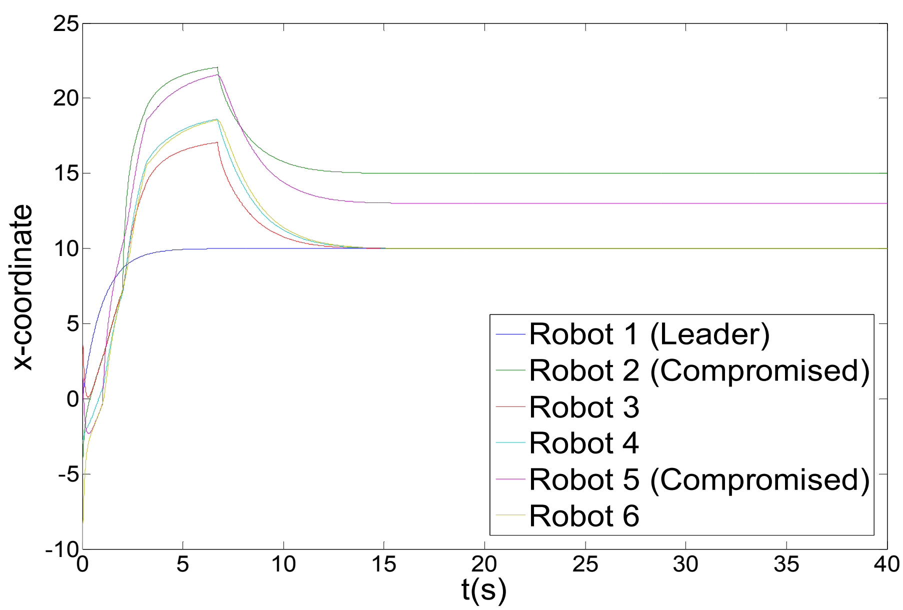

Figure 4.16 Multi-robot formation control result using the proposed secure distributed control algorithm in the presence of two malicious robots

These case studies imply that, if compromised robots exist, the proposed method will be able to gradually mitigate and isolate them, and finally let the well-behaving robots converge to the correct steady state. Furthermore, the convergence is guaranteed as long as the topology satisfies the sufficient conditions of the linear consensus algorithm (we will discuss the robustness of the proposed algorithm in next section).

4.3.5 Simulation Results: Robustness Analysis

The proposed algorithm works with the assumption that the total number of misbehaving agents is less than the connectivity of the network in the worst case. Thus, if we identify one misbehaving agent in the network and we do not know the entire topology of the network, we need to make sure the network is at least 2-connected before we isolate that agent. Otherwise, the network topology after isolation may not satisfy the sufficient conditions for the convergence of the linear consensus algorithm.

As a case study, if Robot 4 is compromised with the communication topology shown in Figure 4.6, then the rest of the well-behaving robots can no longer reach consensus because it has already disconnected the entire network into two parts after the isolation of Robot 4. The corresponding simulation result is shown in Figure 4.17.

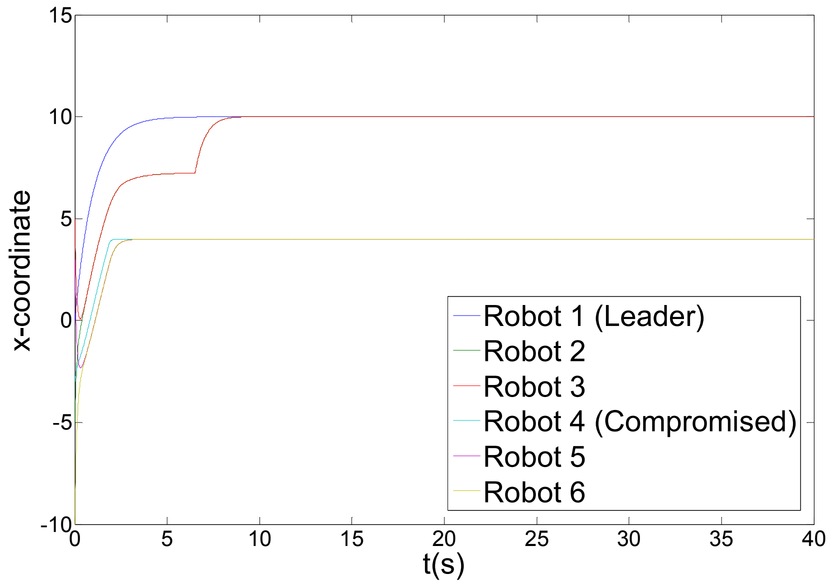

Figure 4.17 Multi-robot formation control result using the proposed secure distributed control algorithm in the presence of one faulty robot in the critical link.

4.4 Experiment Case Studies

To further evaluate the proposed secure distributed control algorithm, we also tested it in our multi-UGV hardware-in-the-loop experimental platform—*i*Space System.

4.4.1 Experimental Platform Setup and Configuration

*i*Space system in the Advanced Diagnosis, Automation, and Control (ADAC) Lab at North Carolina State University (NCSU) is a D-NCS experimental platform composed by several unmanned ground vehicles (UGVs) [64, 65]. It has been implemented to perform research and experiments on D-NCS with hardware-in-the-loop fast-prototyping capabilities.

It allows us to design repeatable experiments under different network settings and system conditions.

The entire system structure of the iSpace system platform is described in Figure 4.18. The experimental platform includes several major components: distributed sensors, distributed actuators, hierarchical and distributed controllers, and communication networks. Specifically, it is composed of the following parts:

• **UGV**: The differential drive UGV used in this testbed has been constructed out of the modular Lego Mindstorms NXT. The wheel radius of the UGV is 0.0286 meters (m) and the distance between wheels is 0.095 m. The maximum speed for the UGV is approximately 0.4 meters per second (m/s) with the default gearing of the NXT motors.

• **Base Station**: The base station is used to combine the distributed sensor information to accomplish the system goal. It is implemented in a high-performance host PC server. In addition to connecting distributed sensors and actuators and implementing the data fusion, the base station can also makes supervisory control decisions.

• **OptiTrack System:** OptiTrack system is a vision system that is capable of tracking all the UGVs in real time. It is an image processing server that can triangulate the positions of the UGVs in 3D space based on eight infrared cameras' observations.

• **Communication Network**: The main control and feedback information flow between the UGVs and the base station is carried over Bluetooth 2.0 with supplementary sensing data (e.g., coordinates of the UGVs) coming from the image processing server through Ethernet.

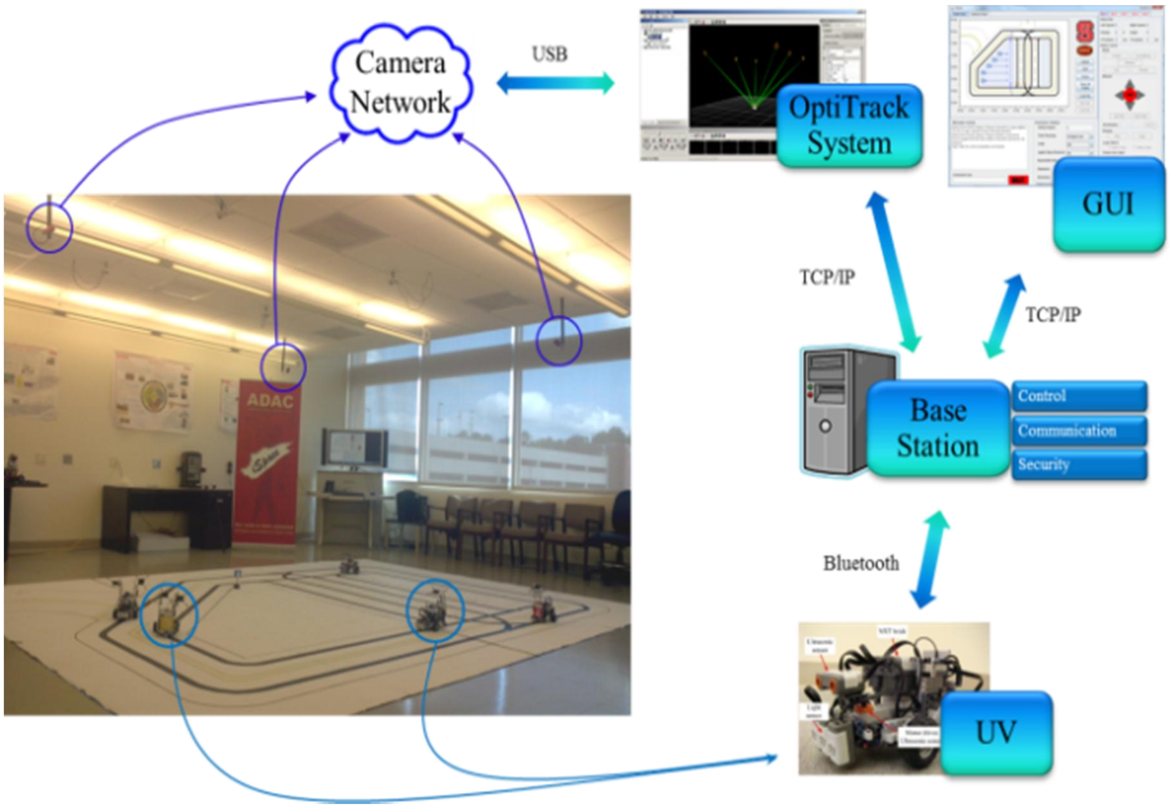

Figure 4.18 System structure of the iSpace system experimental platform

The experiment settings and configurations (e.g., communication topology, malicious inputs and parameters) are the same as those in the simulations described in section V. And the experimental task is still the multi-UGV rendezvous - let all the UGVs reach/assemble the same destination. Similarly, two malicious scenarios (one misbehaving robot and two non-colluding misbehaving robots) and two types of malicious attacks (fault attack and false data injection attack) are both considered and investigated in the experiments.

4.4.2 Experiment Results

The experimental results for the case of a single misbehaving UGV case are shown in Figure 4.19 and 4.20, while experimental results for multiple misbehaving UGVs case are shown in Figure 4.21 and 4.22, respectively. As can be seen, in all four scenarios, all the well-behaving UGVs can reach the correct destination asymptotically, while the compromised UGVs are detected and isolated after a certain period of time. Therefore, we can conclude that the proposed secure distributed control methodology with built-in security mechanism is capable of performing an accurate distributed control computation in the presence of misbehaving agents in the D-NCS and allows all the well-behaving agents to reach the desired consensus state asymptotically.

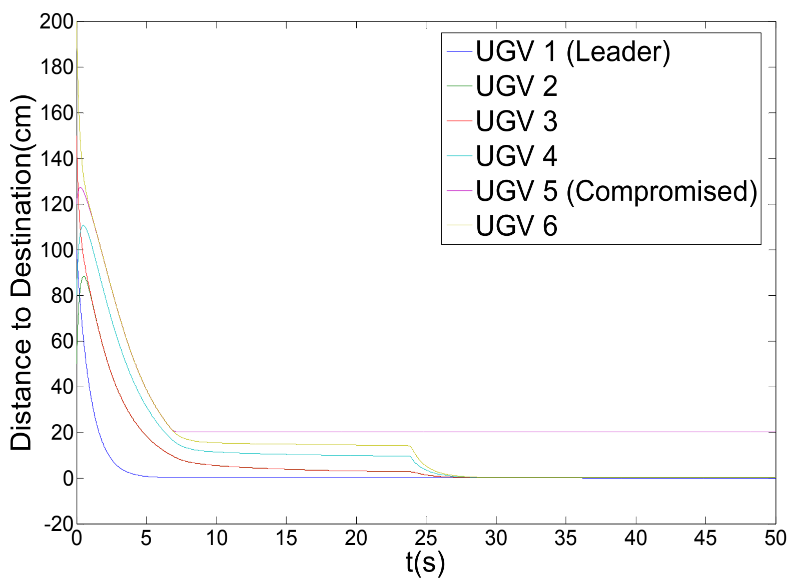

Figure 4.19 Multi-UGV rendezvous experiment results using the proposed secure distributed control algorithm in the presence of one faulty UGV

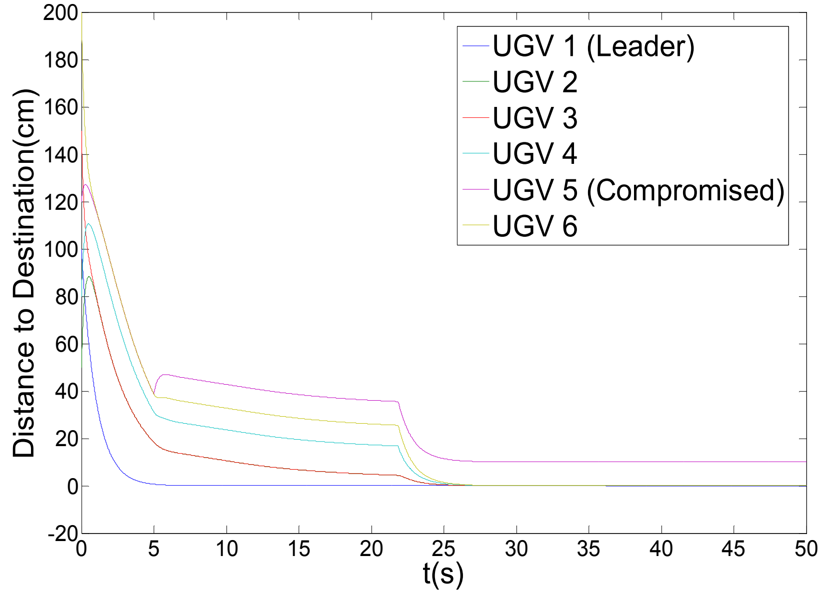

Figure 4.20 Multi-UGV rendezvous experiment results using the proposed secure distributed control algorithm in the presence of one malicious UGV

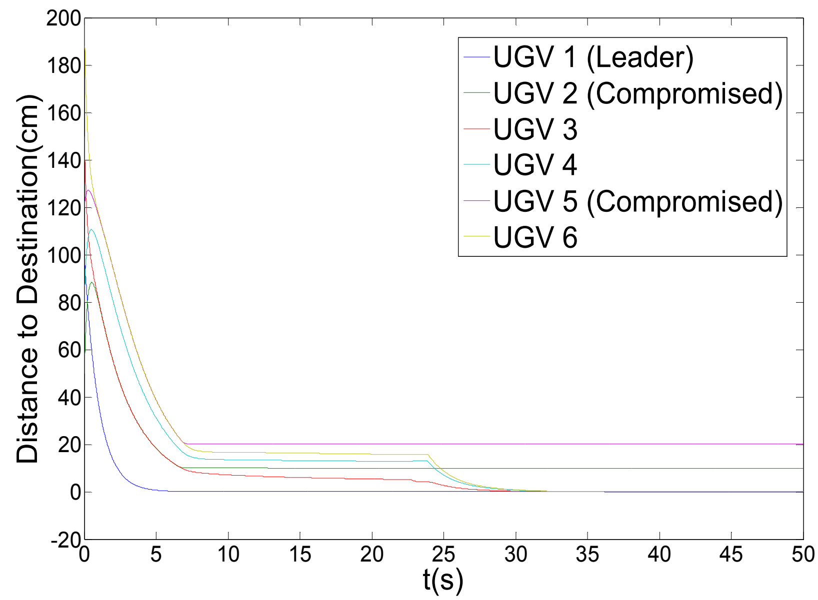

Figure 4.21 Multi-UGV rendezvous experiment results using the proposed secure distributed control algorithm in the presence of two faulty UGVs

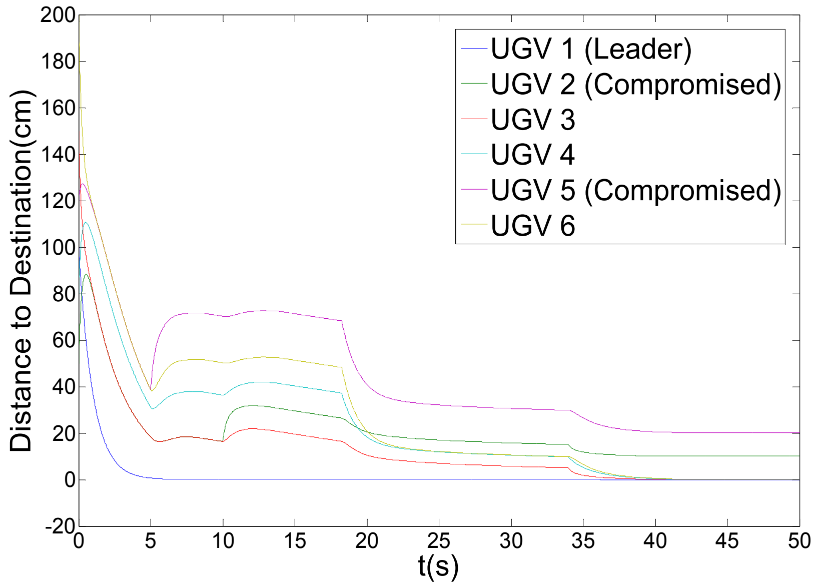

Figure 4.22 Multi-UGV rendezvous experiment results using the proposed secure distributed control algorithm in the presence of two malicious UGVs

CHAPTER 5. RESILIENT DISTRIBUTED CONTROL ALGORITHM FOR LEADERLESS CONSENSUS NETWORK

Regarding 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 algorithm), 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) because the contribution of the misbehaving agent before isolation has already affected the consensus computation.

Thus, in order to eliminate the total contribution from a misbehaving agent in a leaderless consensus network, rather than only 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 time being abnormal (e.g., faulty or malicious) and onwards until isolation.

5.1 Algorithm Description

For the leaderless consensus network, we add a Recovery Phase into our proposed secure distributed control algorithm after the four regular phases. This fifth phase is to remove the impact of misbehaving agents by applying recovery schemes, as shown in Figure 5.1.

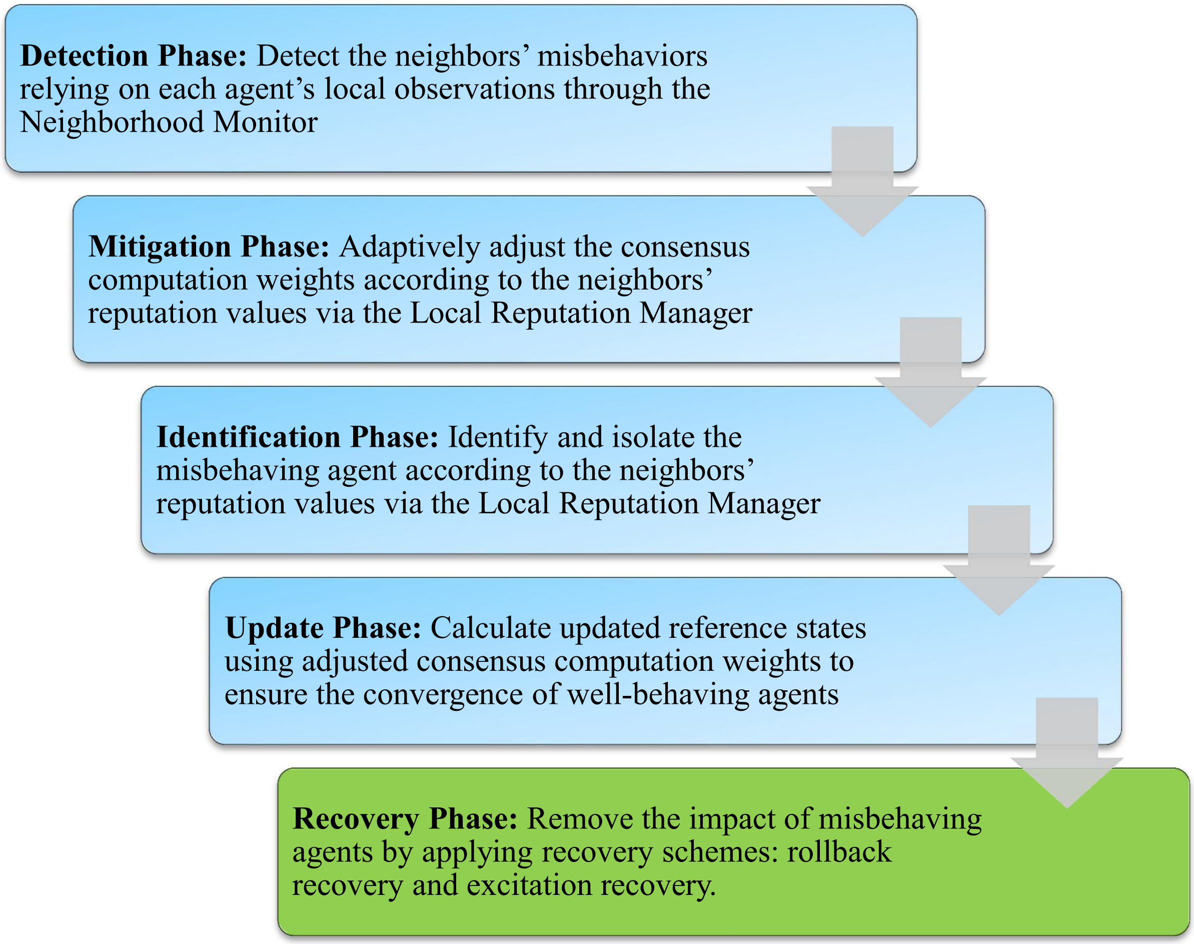

Figure 5.1 Flowchart of the secure distributed control algorithm for the leaderless consensus network

We have developed two types of recovery schemes for different applications: a) Rollback Recovery and b) Excitation Recovery.

5.1.1 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 network since the agent states are easy to reset in these cases.

5.1.2 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 at each agent to compensate the misbehaving agents' effect from the time being abnormal until isolation [66].

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 the neighbors in every iteration.
2) At time $k = k_2$, the abnormal agent is identified as the misbehaving agent and isolated by the secure distributed control methodology, the neighboring agents of the

misbehaving agent apply an external recovery control input $u_{rec}(k)$ such that from time $k = k_2$ to time $k = k_3$,

$$\sum_{k=k_2}^{k_3} u_{rec}(k) = -\sum_{k=k_1}^{k_2} d_{ij}(k)[x_i(k) - x_j(k)]. \tag{33}$$

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 after time n.

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 to the time when the misbehaving agent is first being abnormal, then the final steady state would be weighted average value with well-behaving agents' initial states and the contribution of misbehaving agent before it becomes abnormal, which would be slightly different from accurate consensus value.

Thus, there is a trade-off between the convergence accuracy and algorithm efficiency. Different priority is preferred by setting appropriate record starting time k_1 for different scenarios/applications.

5.2 Simulation Case Studies

To analyze the performance of the proposed secure distributed control algorithm for leaderless consensus network, we apply it to two different distributed control applications respectively: a) multi-robot formation control, and b) wireless sensor network.

While both leader-follower and leaderless networks are tested in the multi-robot formation control case study using the *i*Space Simulator described in Chapter 4 to validate the effectiveness of the proposed method to different network topologies, both faulty agents and malicious agents are considered in the wireless sensor network case study to demonstrate the algorithm's resilience to different types of misbehaving agents in leaderless consensus networks.

Table 5-1 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

5.2.1 Multi-Robot Formation Control: Leader-Follower Case

A representative formation control task – 1-D rendezvous is designed: let all six robots stay in a line in parallel. Their communications topology is shown in Figure 5.2. The initial positions of all the robots are shown in Table 5-1. Since this is a leader-follower consensus network, all the robots should reach the same x-coordinate as the leader's position). Let us set Robot 1 as the leader and Robot 5 as the misbehaving robot becoming faulty (suffering a stopping failure) at time $k = 5$.

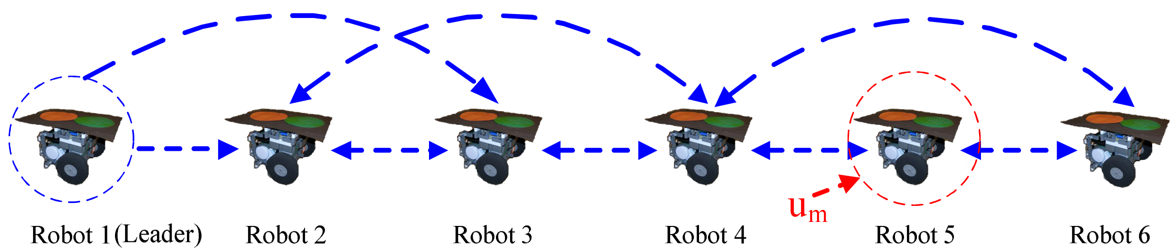

Figure 5.2 The communication topology of the multi-robot system in leader-follower case

We first use the conventional linear consensus algorithm to perform the multi-robot rendezvous task. Figure 5.3 shows the simulation result with a faulty robot. As we can see, only the leader robot reached the desired destination while all the other follower robots stopped at different positions. Thus, the formation control task failed due to the absence of resilience mechanism.

Let us carry out the same task again using the proposed resilient distributed control algorithm. As the simulation results show in Figure 5.4, the misbehaving robot is gradually being isolated and the rest of the normal follower robots achieve the consensus with the leader robot successfully. The convergence is guaranteed as long as the topology satisfies the sufficient conditions of the linear consensus algorithm.

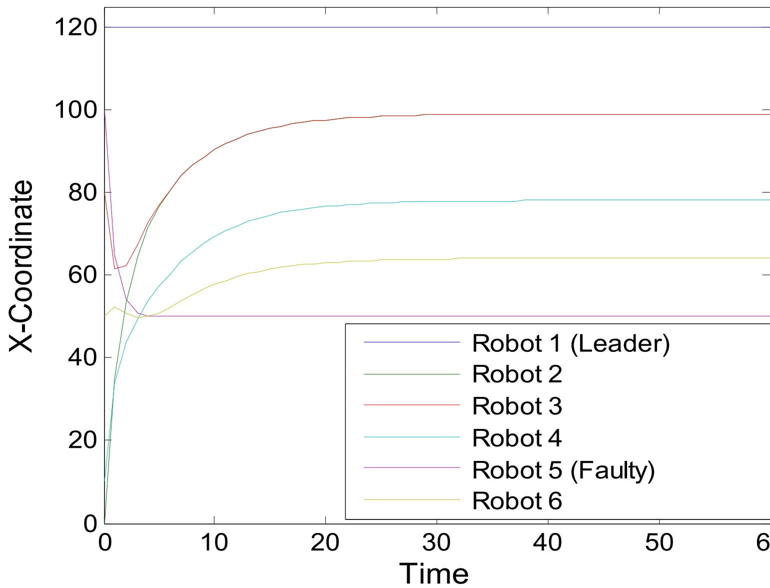

Figure 5.3 The leader-follower multi-robot formation control results in the presence of a faulty robot using conventional linear consensus algorithm

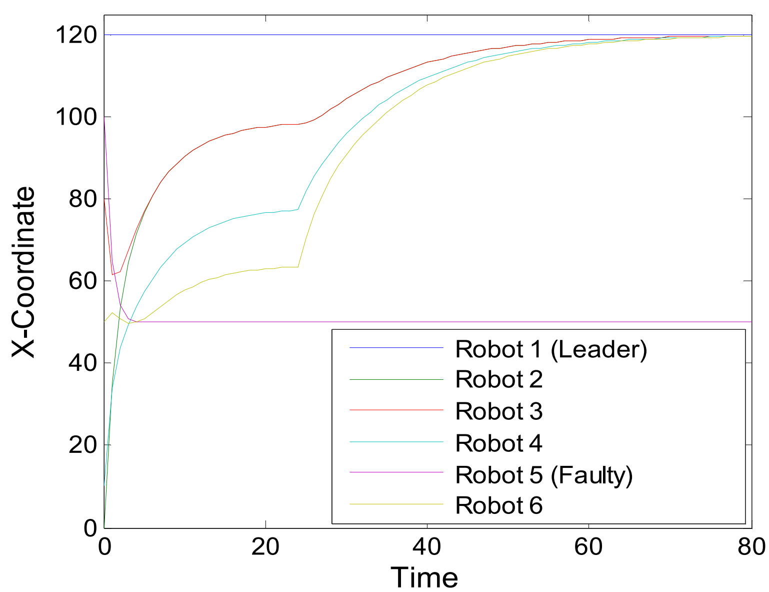

Figure 5.4 The leader-follower multi-robot formation control results in the presence of a faulty robot using the proposed secure distributed control algorithm

5.2.2 Multi-Robot Formation Control: Leaderless Case

The same 1-D rendezvous task setting as the leader-follower case is used in this case study. The only difference is that the communication topology of the robots is set to be fully connected since the leaderless consensus network is applied. Because there is no leader any more, all the robots should reach the same *x*-coordinate which is the average of their initial positions.

The effectiveness of the proposed recovery scheme for leaderless consensus networks is focused and examined in this case study. Excitation recovery is used in this formation control task because of the dynamic nature of the system and the parameters setting are: $k_1 = 0$, $k_2 = 30$, and $k_3 = 35$.

Let us carry out the rendezvous tasks using the proposed secure distributed control algorithm to perform the consensus computation with and without excitation recovery for comparison. As the simulation result shown in Figure 5.5, without excitation recovery, robot 5 drags the entire team away from the desired rendezvous point even it is detected and isolated from the network eventually. However, by applying excitation recovery scheme, the neighboring robots apply the recovery input locally after disconnecting the faulty robot to compensate its previous effect, and then all the well-behaving robots converge at the correct destination asymptotically, which is $x = 52$ in this case. The simulation result shows the trajectories of all the robots in Figure 5.6.

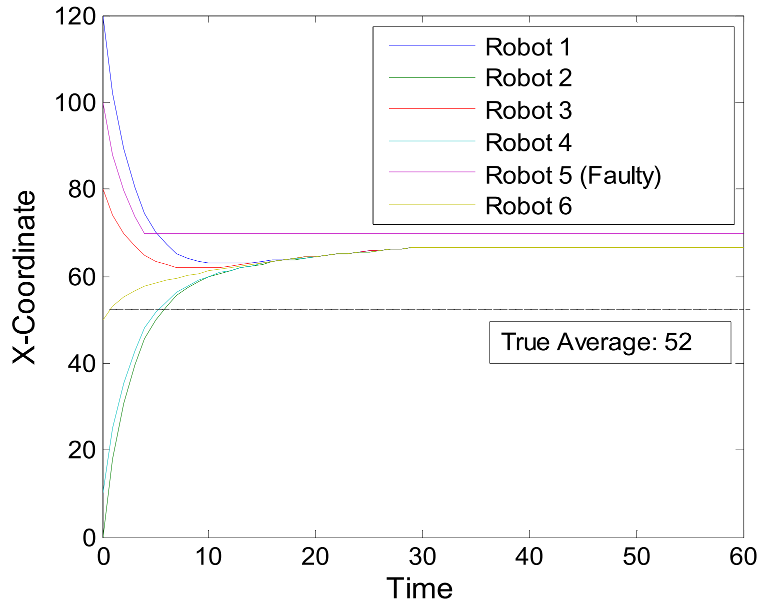

Figure 5.5 The leaderless multi-robot formation control results using the proposed secure distributed control algorithm in the presence of a faulty robot without excitation recovery

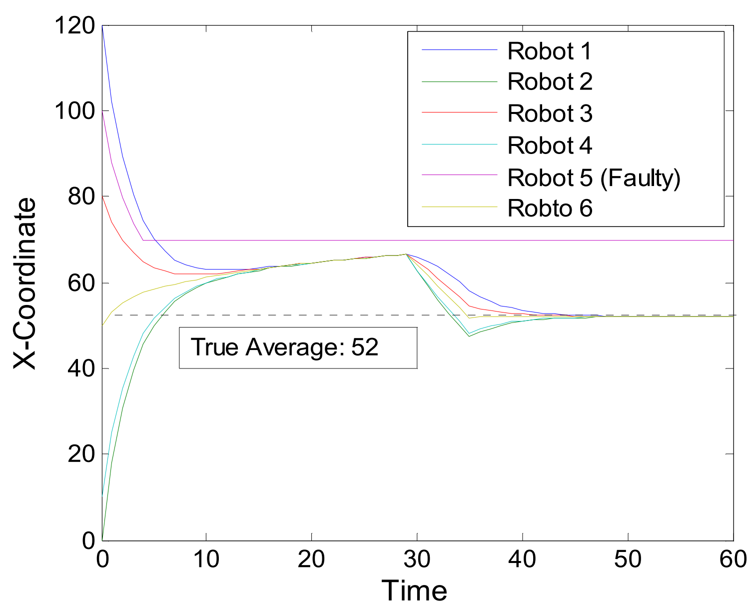

Figure 5.6 The leaderless multi-robot formation control results using the proposed secure distributed control algorithm in the presence of a faulty robot with excitation recovery

5.2.3 Wireless Sensor Network: Faulty Sensor Case

In this task, eight fully connected temperature sensors are used to measure the average temperature of an area, thus the leaderless consensus network topology is used.

The readings of all the temperature sensors are shown in Table 5-2 and let us assume sensor 5 and sensor 6 become faulty (suffer stopping failures) at time $k = 3$. Since the readings of sensors are static, rollback recovery scheme is used in this comparative case study.

Figure 5.7 and Figure 5.8 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 recovery scheme, the convergence value 32.9 ℃ is different from the desired true average of the good sensor readings: 30 ℃ 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.

Table 5-2 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

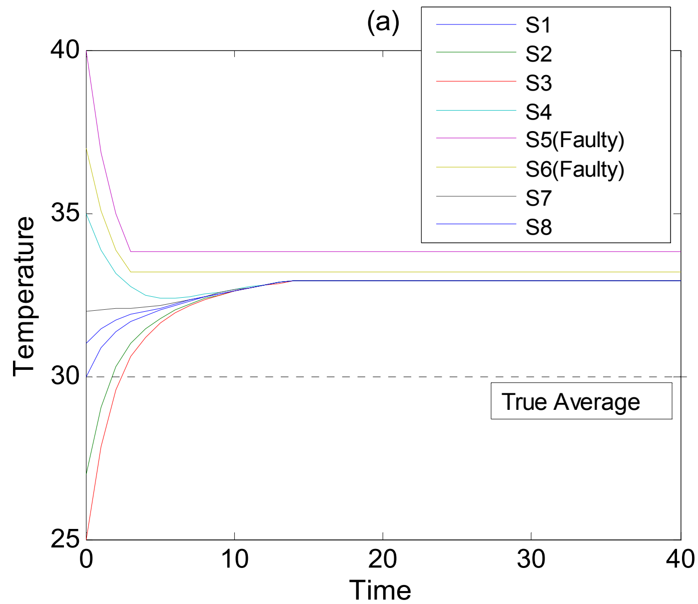

Figure 5.7 The average consensus results using the proposed secure distributed control algorithm in the presence of two faulty sensors without rollback recovery

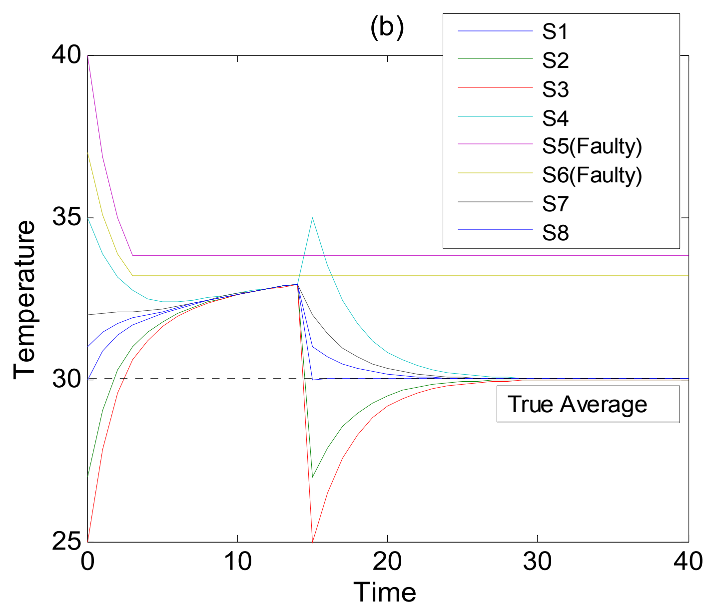

Figure 5.8 The average consensus results using the proposed secure distributed control algorithm in the presence of two faulty sensors with rollback recovery

5.2.4 *Wireless Sensor Network: Malicious Sensor Case*

In this case study, sensor 5 and sensor 6 start malfunctioning from time $k = 3$. Let us set sensor 5 suffers malicious cyber attack with $u_{5,M}[k] = 1$ and sensor 6 suffers malicious cyber attack with $u_{6,M}[k] = 2$. All the other settings remain the same as the faulty case above. Rollback recovery scheme is still used in this case study.

Figure 5.9 shows the simulation results of the average consensus in the presence of two malicious sensors without recovery schemes. The convergence value 35.8 ℃ is far away from the desired true average temperature 30 ℃ due to the malicious effect of two compromised sensors. However, when the rollback recovery scheme is added, as shown in Figure 5.10, the correct average value is asymptotically reached again.

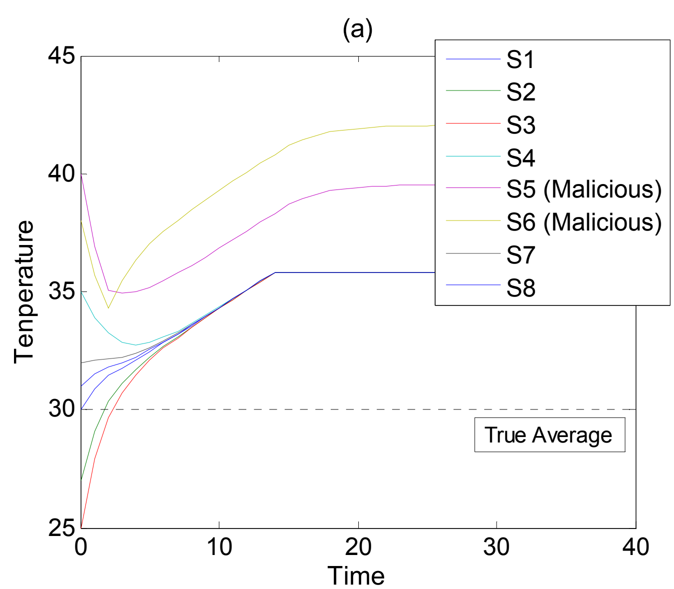

Figure 5.9 The average consensus results using the proposed secure distributed control algorithm in the presence of two malicious sensors without rollback recovery

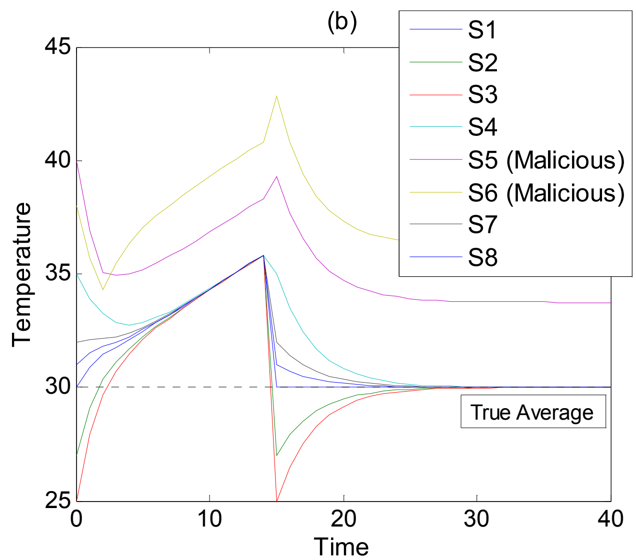

Figure 5.10 The average consensus results using the proposed secure distributed control algorithm in the presence of two malicious sensors with rollback recovery

These case studies implies that, if misbehaving agents exist, the proposed method will be able to gradually mitigate and isolate them, even compensate their effect to the network, and finally let the well-behaving agents converge to the correct steady state, in both leader-follower and leaderless consensus networks.

CHAPTER 6. TRADE-OFF MODELING FOR PERFORMANCE AND SECURITY IN NCS

The traditional NCS designed without security protection are vulnerable to various security attacks. There is a growing demand of efficient and scalable Intrusion Detection Systems (IDS) embedded in the NCS. Furthermore, using security in an NCS gives rise to many topics like network architecture to support security for NCS, the performance assessment for NCS with security, etc. Cardenas et al. [34] gave an overview of security issues in Cyber Physical Systems (CPS), identified and defined the problem of secure control and proposed a set of challenges that need to be addressed to improve the survivability of CPS. Mukherjee [67] established a Criticality Response Modeling (CRM) framework to ensure the networked control system has criticality-awareness – the ability of the system to respond to unusual situations. Tsang and Kwong [68] proposed an efficient and biologically inspired learning model for multi-agent IDS utilized in the network infrastructures of industrial plants. Creery and Byres [69] presented methods to determine and reduce the vulnerability of NCS to unintended and malicious intrusions for an industrial plant. Xu et al. [70] developed core architecture to address the collaborative control issues of distributed device networks under open and dynamic environments by adopting policy-based network security and XML technologies. Gupta et al. [71, 72] characterized the NCS application on the basis of security effect on NCS performance and mapped the added security features to additional time delay in the system to show this trade-off for a wireless NCS robot path tracking application.

Although NCS with security from a control system perspective is still in its infancy, many NCS have been well protected by security mechanisms as stated above. However, the added security features may sacrifice system dynamic performance due to limited system resources. The impact of the security mechanisms on the system dynamic performance has not been addressed thoroughly. Security requirements are often in conflict with other performance requirements, like real-time dynamic performance due to limited system resources and extra time delay imposed by security additions.

Motivated by the above analysis, this chapter investigates the effect of the addition of security mechanisms in NCS. In this chapter, a secured networked DC motor system is used to illustrate the proposed concept. The DC motor dynamics is modeled by differential equations, while the security mechanisms are modeled by discrete events. A trade-off model for the dynamic performance and security in this secured NCS is established. The construction of the trade-off model involves the development of: (1) a set of metrics to quantitatively measure the DC motor closed-loop dynamic performance and security levels, and (2) a trade-off objective function incorporating both the performance and the security metrics simultaneously.

6.1 Problem Formulation

A secured networked DC motor system is depicted in Figure 6.1, where the sensor and the actuator are time-driven with the same synchronized sampling period. The controller is event driven as the controller signal is calculated as soon as the sensor data is available on the controller side. The security mechanism is also event driven by the detection of adversary attacks. This secured networked DC motor system can be divided into four parts: (a) The local DC motor system (including the actuator, the DC motor and the sensor); (b) The networked controller; (c) The security mechanism; and (d) The communication network [73, 74]. Each component is described in the following sections.

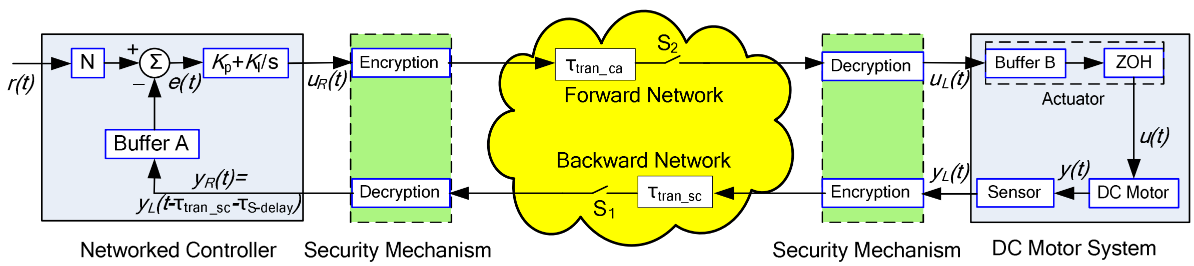

Figure 6.1 The structure of the secured networked DC motor system

6.1.1 The Local DC Motor System

The electromechanical dynamics of the DC motor is described as:

$$\dot{i}_a = -\frac{R}{L} i_a - \frac{K_b}{L} \omega + \frac{1}{L} u \, , \tag{34}$$

$$\dot{\omega} = \frac{K_t}{J} i_a - \frac{B}{J} \omega \ , \tag{35}$$

where i_a is the armature winding current; ω is the rotor angular speed; R is the armature winding resistance; L is the armature winding inductance; K_b is the back-electromotive-force

(EMF) constant; K_t is the torque constant; J is the system moment of inertia; B is the system damping coefficient. u is the armature winding input voltage.

Let $x = [x_1, x_2]^T = [i_a, \omega]^T$, the system output is the rotor angular speed $y = \omega$, the DC motor system can be expressed by the state-space equations:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -\frac{R}{L} & -\frac{K_b}{L} \\ \frac{K_t}{J} & -\frac{B}{J} \end{bmatrix} x(t) + \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} u(t) \tag{36}$$

$$y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} x(t) \tag{37}$$

For simplicity, we assume that the local controller simply converts the control voltage data sent from the networked controller into a PWM signal to drive the DC motor. Thus $u_L(t) = u(t)$, where $u_L(t)$ is the input of the local controller. Besides, we denote $y_L(t)$ as the output of the sensor measuring the system's rotor angular speed y. Thus, $y_L(t) = y(t)$.

6.1.2 *The Networked Controller*

The system uses a PI controller to control the DC motor's speed:

$$u_R(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau \tag{38}$$

$$e(t) = r(t) - y_R(t) \tag{39}$$

where $u_R(t)$ is the control signal from the networked controller to the local controller; $r(t)$ is the referenced speed; $y_R(t)$ is the measured signal received from the sensor. K_p and K_i are the proportional and integral gain for the PI controller to be designed.

6.1.3 The Security Mechanism

This section will focus on the confidentiality aspect of security service using secret key cryptography. The Data Encryption Standard (DES) and the Advanced Encryption Standard (AES) are two symmetric ciphers which are considered secure for wireless systems. DES is one of the most important classical cryptosystems in the history of cryptography. 3DES is very similar to DES and requires three keys (encryption-decryption-encryption) to provide stronger security than DES. Consequently, 3DES is 3 times slower than DES. AES, a successor of DES, works with 128 bits of block size with key sizes 128 or more. As ECB (Electronic Code Book) is considered the fastest mode of operation, it is used commonly in real time systems. Therefore, in this paper, secret key algorithms DES, 3DES and AES [75] with ECB mode are embedded in the NCS to encrypt and decrypt the information flow between the networked controller and the DC motor system [76].

With different encryption algorithms, the security mechanism is modeled as a discrete event system [77], which has three parts: security sensors, security decisions and security actuators.

- ***Security Sensors***: to receive the sensor data to check if the system is under attack.
- ***Security Decisions***: to process the sensor data, make security decisions (decide which encryption algorithm to use).
- ***Security Actuators***: to encrypt the data packet and result in additional time delay in the system.

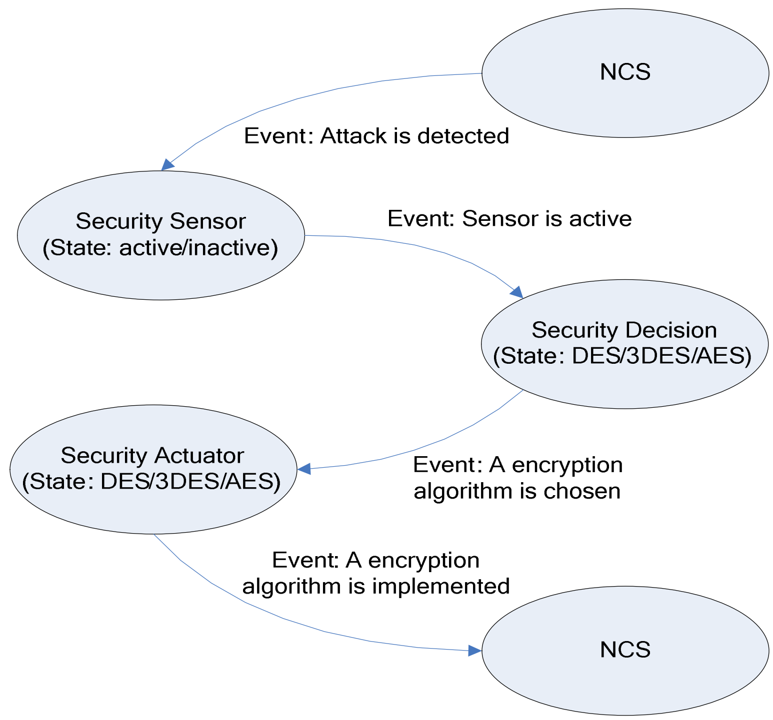

Figure 6.2 State flow of the security mechanism based on discrete events

The state flow of the security mechanism is shown in Fig. 6.2. There are four steps in this discrete event process:

1) The security sensors read Boolean status variables from the input at each update and the event that the input is true can trigger its state to active;
2) When the sensor status is active, a broadcast event is sent to the security decision state;
3) An event of changing the security decision state can trigger the security actuator state change from one to another;
4) An event of changing the security actuator state is sent to NCS.

6.1.4 The Communication Network

The data packets in the networked DC motor system usually suffer time delay during the transmissions. So we let τ_{sc} denote the sensor-to-controller time delay and τ_{ca} denote the controller-to-actuator delay. Then the total Round Trip Time (RTT) delay can be described as:

$$\tau_{RTT} = \tau_{sc} + \tau_{ca} \tag{40}$$

And τ_{sc} and τ_{ca} can be described as:

$$\tau_{sc} = \tau_{tran\_sc} + \tau_{S-delay}, \tag{41}$$

$$\tau_{ca} = \tau_{tran\_ca} + \tau_{S-delay}, \tag{42}$$

where τ_{tran_sc} and τ_{tran_ca} are the transmission delays; $\tau_{S-delay}$ is the extra time delay imposed by security additions.

Thus, the signal relationship between the networked controller and the local controller and sensors are:

$$u_L(t) = u_R(t - \tau_{ca}) \tag{43}$$

$$y_R(t) = y_L(t - \tau_{sc}) \tag{44}$$

6.2 The Performance-Security Trade-off Model

6.2.1 The Performance Metric

NCS may have different performance requirements for different applications. This research focuses on the main impact of security mechanisms on the real-time dynamic performance of the system.

We use three measures 1) the mean square error J_1, 2) the percent overshoot J_2 and 3) the rising time J_3 to evaluate the system dynamic performance:

$$J_1 = \begin{cases} 0 & , \quad MSE \le MSE^* \\ \dfrac{(MSE - MSE^*)}{MSE^*} & , \quad MSE > MSE^* \end{cases}, \tag{45}$$

$$J_2 = \begin{cases} 0 & , \quad P_{OS} \le P_{OS}^* \\ \dfrac{(P_{OS} - P_{OS}^*)}{P_{OS}^*} & , \quad P_{OS} > P_{OS}^* \end{cases}, \tag{46}$$

$$J_3 = \begin{cases} 0 & , \quad t_s \le t_s^* \\ \dfrac{(t_s - t_s^*)}{t_s^*} & , \quad t_s > t_s^* \end{cases}, \tag{47}$$

$$MSE = \frac{\sum_{K=1}^{N} e^2(KT)}{N} = \frac{\sum_{K=1}^{N} [r(KT) - y(KT)]^2}{N}, \tag{48}$$

where MSE is the mean square error of the speed tracking. MSE^* is the nominal mean squared error; P_{OS} is the percentage overshoot of the controlled signal value and $\mathrm{t_S}$ is the setting time of the system. P_{OS}^* and t_S^* are the nominal values of percentage overshoot and setting time. They are the nominal performance criteria that the system should achieve without time delay in the control loop.

The corresponding cost function can be defined as follows:

$$J = w_1 J_1 + w_2 J_2 + w_3 J_3, \tag{49}$$

where w_1, w_2 and w_3 are the weight factors. They are used to specify the relative significance of J_1, J_2 and J_3 on the overall system performance and map to a scalar measure.

Then, the cost function J is inversed and normalized to define a system performance metric P:

$$P = \frac{J_{\min}}{J},\ 0 \le P \le 1, \tag{50}$$

where J_{min} is the minimum J that can be obtained.

6.2.2 The Security Metric

Existing qualitative metrics classify various security mechanisms to several discrete levels, such as low, medium, and high. Security mechanisms in the higher level can provide better protection than lower levels. However, it is impossible to compare security mechanisms within the same security level. Furthermore, qualitative metrics are too coarse for fine control of the trade-off between NCS dynamic performance and security. Thus, a quantitative metric that generates a security strength value for each security mechanism is used in this paper, and hence is more suitable for quantitative comparison of the security strength of any two security mechanisms [78].

Without considering any shortcut attacks, brute force attack is the only way used to crack the encryption key in this paper. For example, a DES cipher with a key length of 56 bits has 2^{56} possible key combinations. Assuming unit complexity for testing one key, the worst-case complexity involved in cracking this 56-bit DES cipher is 2^{56}. With this assumption, the security level of an encrypted message frame is decided by its encryption key length. Thus, a security measure with respect to brute force attacks is described to be $S(N) = log_2(N)$, where N is the encryption key length. Let a message consist of n frames with encryption key length N_i bits for frame $i = 1,2, \dots, n$, and assume that every frame of this message is equally important to decrypt the message, then a reasonable security measure for the whole message is the mean of the security levels achieved by the individual frames [79]. Therefore, the security level is measured by the vulnerability of encryption algorithms to brute force attackers, which denotes the security metric S:

$$S = \frac{1}{nS_{\max}} \sum_{i=1}^{n} \log_2 N_i \ , \ 0 < S < 1, \tag{51}$$

where $S_{max} = max(log_2 N_i)$.

6.2.3 Performance and Security Requirements

With the defined performance metric P and security metric S, the system requirements can be formulated quantitatively.

The dynamic performance and security operations depend on the operating requirements. For example, the DC motor needs to reach a reference speed in a certain time within a certain overshoot and a certain tracking error, while satisfying the required security level. This could also be called as the minimum requirements that need to be satisfied:

$$P > P_o \, , \ S > S_o \, , \tag{52}$$

where P_o is the operating requirements of the system dynamic performance; S_o is the operating requirement of the system security level.

Secondly, the control system should have a feasible operating region. Out of this region, the system may have oscillation, even going out of control. Thus, P and S must be within this stable region. This could be called as the stabilization requirements that need to be satisfied:

$$P < P_s \,,\; S < S_s \,, \tag{53}$$

where P_S is the stabilization requirement of the system dynamic performance; S_S is the stabilization requirement of the system security level.

By combining these two parts together, the overall system dynamic performance and security requirements cab be described as follows:

$$P_o < P < P_s \,,\; S_o < S < S_s \,, \tag{54}$$

6.2.4 The Trade-off Objective Function

The dynamic performance and security metrics allow us to quantitatively calculate how much protection a security mechanism can provide and how much dynamic performance will be reduced by using that security mechanism. Hence, we can make the trade-off between dynamic performance and security by adjusting the system control inputs and security parameters, within the boundary of the system requirements.

When both the dynamic performance and security requirements are satisfied, the system is capable of using the available resources for improving the performance and/or the security. A trade-off objective function between the performance metric and the security metric can be formulated as a utility function:

$$\text{Max } U = w_{u1}P + w_{u2}S$$

$$\text{with } P_o < P < P_s,\ S_o < S < S_s, \tag{55}$$

where w_{u1} and w_{u2} ($w_{u1} + w_{u2} = 1$) are two weighting factors representing the preferences on performance and security, respectively.

With the objective function U, the system can compute and choose the best security parameters together with optimal control inputs according to the system requirements to achieve the best trade-off between dynamic performance and security.

6.3 Simulation Case Study: Networked DC Motor System

We developed a Simulink based test-bed to evaluate the trade-off model in the secured networked DC motor system in real-time with utility constraints and varying inputs. The DC motor system was embedded with a networked PI controller to control the speed of the motor over a network.

6.3.1 Testbed Description

As shown in Figure 6.3, the test-bed is implemented in Matlab Simulink according to the system description in Section 6.1. The DC motor dynamics is simulated using equations (36) and (37). A wireless network environment with random time delay and packet drop rate was simulated such that each desired network parameter can be user controlled. The network security algorithms were implemented on this test-bed to study the effect of network security additions. In DES/3DES and AES, the total time for encryption as well as decryption depends upon the encryption key length between controller and the plant. Experiments were performed on a Linux Machine (Kernel 2.4.18-3 i386) to find out the mean processing time for DES, 3DES and AES, and the statistic results are shown in Table 6-1. While the DC motor system dynamics is implemented in the continuous domain, the security mechanism is simulated as a discrete event system by the State Flow in the Simulink, which is also a subsystem of the encryption/decryption blocks in Figure 6.3.

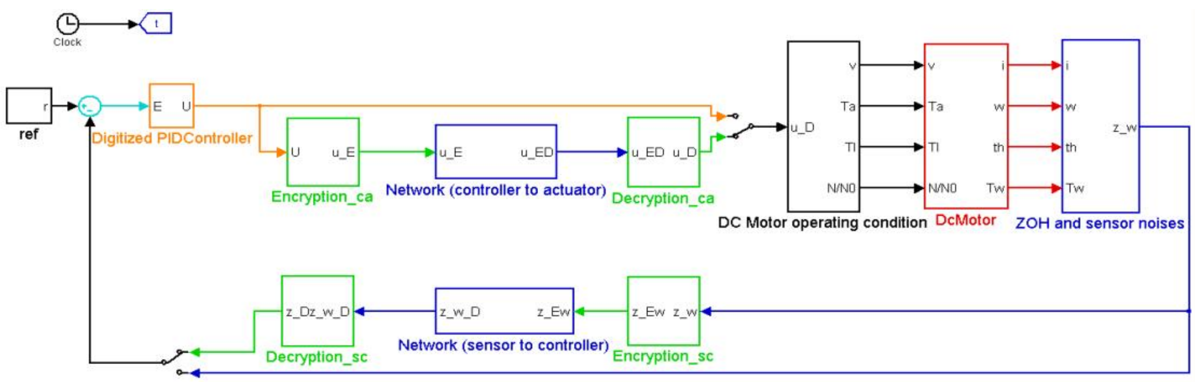

Figure 6.3 Simulink based test-bed for the secured networked DC motor system

Table 6-1 Induced Time-Delay with respect to Security Algorithms

Encryption Algorithms	DES	3DES	AES
Processing Delay (ms)	21.1	54.1	68.5

6.3.2 Test Conditions

The parameters of the DC motor are listed in Table 6-2. To achieve the desired closed loop performance, the nominal gains have been tuned to be $K_p = 1$ and $K_i = 0.01$. The sensor and the output data are sampled with sampling period $h = 1$ ms by considering the buffer size equals to 1.

The network transmission delay on the Internet is random in nature. Generally speaking, it is in the order of a few hundreds of millisecond, typically from 100ms to 600ms on the Internet. Therefore, $\tau_{\mathrm{tran\_sc}}$ and $\tau_{\mathrm{tran\_ca}}$ is set to be in the range of [100ms, 600ms]. The additional time delay imposed by the security additions, $\tau_{\mathrm{S-delay}}$, is simulated according to the statistic results of the processing delay for DES, 3DES and AES, which have been shown in Table 6-1.

Table 6-2 The Parameters of The DC Motor

J	Motor Moment of Inertia	$3.53 \times 10^{-5} kg \cdot m^2$
L	Motor Winding Inductance	$2.88 \times 10^{-3} H$
R	Motor Winding Resistance	6.43Ω
K_t	Electric Torque Constant	$2.55 \times 10^{-2} N \cdot m/A$
K_b	Back-EMF Gain Constant	$2.55 \times 10^{-4} V \cdot s/rad$

The performance metric specifications for the networked DC motor speed control system are:

- Mean square error of the speed tracking: $MSE^* \leq 0.05$;
- Percentage overshoot: $P_{OS}^* \leq 15\%$;
- 5% settling time: $t_S^* \leq 5s$;
- $w_1 = w_2 = w_3 = 1/3$.

The security metric specifications for the networked DC motor speed control system are:

- No security: $N = 1$;
- DES: $N = 56$;
- 3DES: $N = 112$;
- AES: $N = 128$;
- $S_{max} = log_2 128$.

6.3.3 Simulation Results: Static Case

Our experimental task is to drive the networked DC motor to a reference speed 50 rad/s with different encryption algorithms. Figure 6.4 shows the system dynamic performance and security levels using different encryption algorithms. The security level increases while the dynamic performance level decreases with respect to different encryption algorithms. A paired t-test was conducted to compare the mean dynamic performance and security levels of different encryption algorithms using 10 samples in each algorithm.

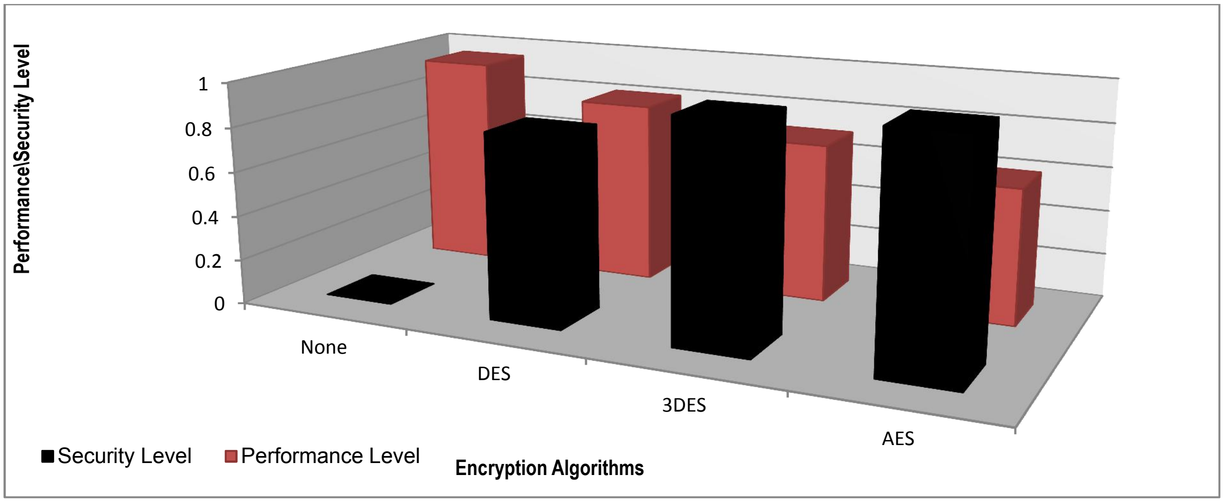

Figure 6.4 The system performance and security levels using different encryption algorithms

As we can see in Table 6-3, all the confidence levels do not cover zero, all of these conditions conclude that there is a statistically significant difference between the performance and security levels for each pair of compared algorithms at 5% level of significance. Therefore, as we increase the security level, statistically, there is a significant reduction in the performance level. Table 6-4 shows the averaged amount of reduction in the performance levels of different encryption algorithms as compared to the condition without encryption algorithms. This value was computed as:

$$\textbf{\% Reduction} = \frac{P_{none} - P_{DES/3DES/AES}}{P_{none}} \times 100\% . \tag{56}$$

Table 6-4 suggests that the reductions in performance level are practically significant due to different encryption algorithms. This means encryption algorithms do have significant effect on NCS performance and our model can show this trade-off quantitatively clearly.

Table 6-3 Paired T-Test Comparing the Mean Performance and Security Levels of Different Encryption Algorithms

Pair of Encryption Algorithms	Security Level Difference	95% CI of Performance Level Difference
(None, DES)	0.8296	[-0.1438, -0.1316]
(DES, 3DES)	0.1429	[-0.1156, -0.0954]
(3DES, AES)	0.0275	[-0.1295, -0.0819]

Table 6-4 Percentage Reduction in Performance Level using Different Encryption Algorithms

Encryption Algorithm	DES	3DES	AES
% Reduction	14.42%	25.48%	36.55%

6.3.4 Simulation Results: Dynamic Case

Let us carry out the same task above while satisfying the following dynamic security level (the dash line in Figure 6.5). There are 10 different security requirements in 10 different consecutive time periods. In order to fulfill the requirement, different encryption algorithms are applied in 10 time periods in the simulation (listed in Table 6-5). After applying the security mechanisms above, the actual security level achieved in the system is shown in Figure 6.5 in solid line. Since the actual security level is above the minimum required security level, the comparison indicates that the security mechanisms applied in the simulation have satisfied the minimum system security requirements.

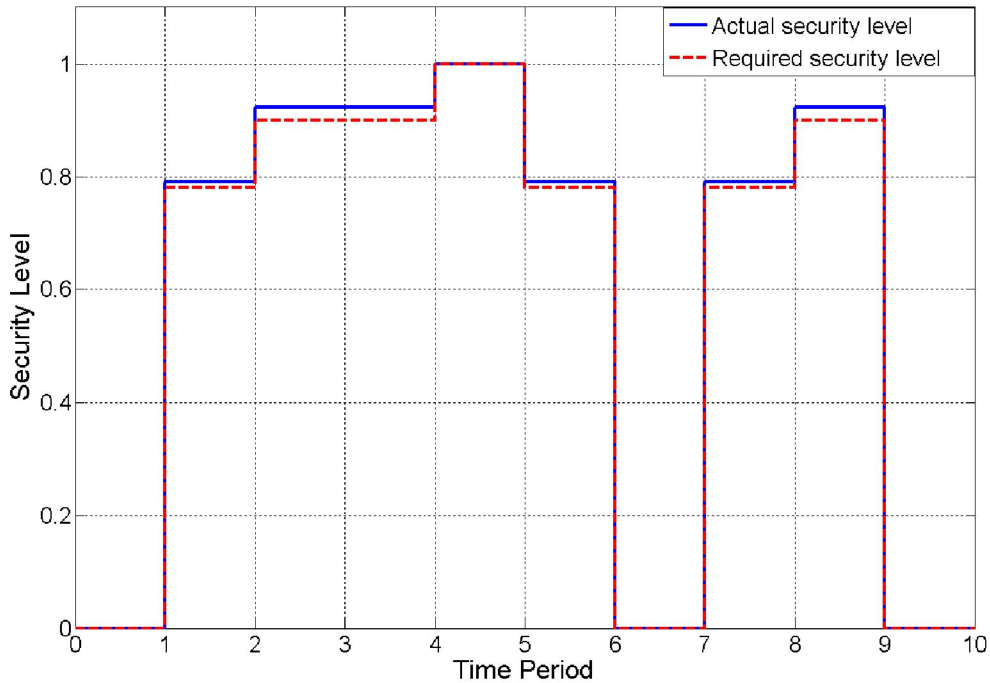

Figure 6.5 The comparison between the required security level and actual security level of the secured networked DC motor system

Table 6-5 The Security Mechanisms Configuration

Time Period	Security Algorithm	key Length
1	None	1
2	DES	56
3	3DES	112
4	3DES	112
5	AES	128
6	DES	56
7	None	1
8	DES	56
9	3DES	112
10	None	1

While maintaining the required minimum security level, we ran the simulation through the 10 time periods and measured the corresponding system performance and security levels. Figure 6.6 shows the averaged system performance and security levels in 10 consecutive time periods based on 10 repeated simulations. As we can see, the trade-off between the system performance and security of the secured networked DC motor system is presented clearly. Apparently, the trade-off model and its relevant metrics show the trade-off between performance and security of the secured networked DC motor system very well, which demonstrates that the proposed trade-off model provides a satisfactory quantitatively performance and security measurements for the NCS.

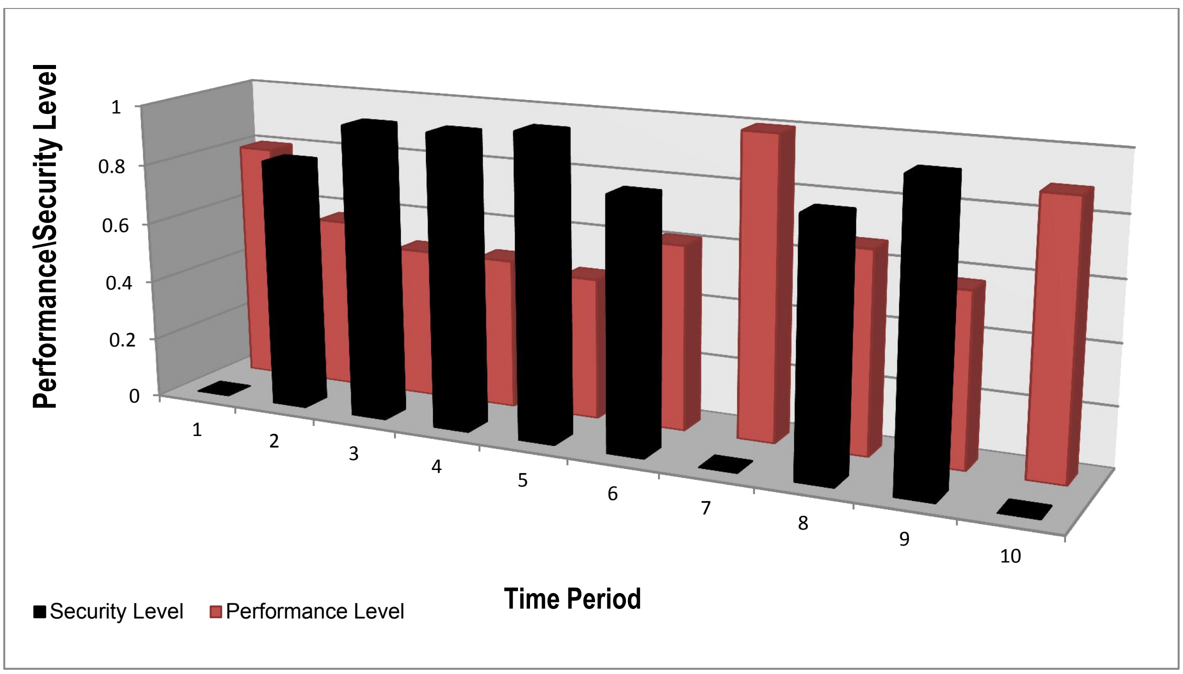

Figure 6.6 The trade-off between the system performance and security of the secured networked DC motor system

CHAPTER 7. PERFORMANCE AND SECURITY TRADE-OFF OPTIMIZATION IN NETWORKED DC MOTOR SYSTEMS

A trade-off model on NCS has been described in the previous chapter [37]. A set of quantitative performance and security metrics have also been developed and combined in a tradeoff objective function. However, how to achieve the optimized balance between performance and security on this trade-off model has not been addressed.

In this chapter, in order to optimize this trade-off on NCS, we will use Evolutionary Algorithms (EAs) and more specifically the Coevolutionary Genetic Algorithm (CGA). As a fast-developing optimization algorithm, CGA is an extension of conventional evolutionary algorithms. It models an ecosystem consisting of two or more species. Multiple species in the ecosystem coevolve and interact with each other and result in the continuous evolution of the ecosystem [80, 81]. CGA has been gradually obtaining extensive applications in different areas [82]. For example, Chen introduced a new framework of electricity market analysis based on Coevolutionary Computation (CCEM) [83]; Careri used CGA to solve the Strategic Bidding problem in a Day-Ahead Market [84]; Skolicki discussed proactive security with focus on emerging security patterns in the context of CGA [85]. CGA has demonstrated that it outperforms a sequential GA on many hard problems. Motivated by the above applications, this chapter presents a framework of performance-security trade-off optimization for NCS based on CGA on a networked DC motor speed control system as a typical example of NCS. This chapter will find the Nash equilibrium based on CGA for the trade-off model on NCS and evaluate the performance of CGA as well.

7.1 Trade-off Model for Networked DC Motor system

7.1.1 The Performance Metric

NCS may have different performance requirements for different applications. The real-time dynamic performance is our major concern for this networked DC motor system. We use the mean square tracking error to evaluate the system's dynamic performance. The corresponding cost function is defined as:

$$P = \begin{cases} 0 & , \quad when \ \overline{E} \le \overline{E}^*, \\ \dfrac{(\overline{E} - \overline{E}^*)}{\overline{E}^*} & , \quad when \ \overline{E} > \overline{E}^*, \end{cases} \tag{57}$$

where $\overline{E} = \dfrac{\sum_{k=1}^{K} e^2(k)}{K} = \dfrac{\sum_{k=1}^{K} [r(k) - y(k)]^2}{K}$, (58)

K is the total number of sampling points, $\bar{E}$ is the mean square error of the speed tracking, $\bar{E}^*$ is the nominal mean squared error, which is the nominal performance criterion that the system should achieve without time delay in the control loop.

7.1.2 The Security Metric

The security metric of the system is composed of two parts: one is the vulnerability of the encryption algorithms to malicious attacks, while the other is the system oscillation level.

A vulnerability measure of the encryption algorithms with respect to brute force attacks is described in literatures as $S(N) = log_2(N)$, where N is the encryption key length [78] [79]. Assuming that the frame of messages in the sensor-to-controller channel and the controller-to-actuator channel are equally important to the entire system, a reasonable vulnerability measure of the system is the mean of the vulnerability values achieved by these two communication channels. A better security of the system also means the system should have a better tolerance of system perturbation caused by malicious attacks. And we know that the system will have better system robustness against the malicious attacks if it has a smaller oscillation, which is positively proportional to the system mean square tracking error – the performance metrics P. Thus we add a system oscillation factor in the security metric.

Therefore, the excellence of the security setting is measured by both the vulnerability of the encryption algorithms (to brute force attackers in this paper), and the system oscillation level, and is denoted as:

$$S = \alpha \frac{2}{\log_2 L_{sc} + \log_2 L_{ca}} + \beta P, \tag{59}$$

where α and β are the weighting factors for normalization. These two weights factors are also allowed to adjust according to the preference/need of the system security. Normally, they should be chosen based on the real system settings and requirements using heuristics.

7.1.3 *The Trade-off Objective Function*

Combining all the dynamic and metric equations above, we can clearly find out that there exist nonlinear mappings from K_P, K_I, L_{sc} and L_{ca} to the system performance metric P and the security metric *S*. Thus, we can formulate the P and S as the functions of K_P, K_I, L_{sc} and L_{ca}, and are represented by the function *f(·)* and the function *g(·)*, respectively:

$$P = f(K_p, K_I, L_{sc}, L_{ca}), \tag{60}$$

$$S = g(K_p, K_I, L_{sc}, L_{ca}), \tag{61}$$

where K_P and K_I are the system dynamic performance parameters, L_{sc} and L_{ca} are the security parameters described in previous sections. *f(·)* is the nonlinear mapping from K_P, K_I, L_{sc} and L_{ca} to *P*, while *g(·)* is the nonlinear mapping from K_P, K_I, L_{sc} and L_{ca} to *S*.

With a given operating specification, we need to add the following constraints:

$$P_{\min} < P < P_{\max}, S_{\min} < S < S_{\max}, \tag{62}$$

where P_{min} and P_{max} are the operating requirements of the system dynamic performance, S_{min} and S_{max} are the operating requirements of the system security level, Figure 7.1 shows the conceptually admissible operating region of the trade-off model with the constraints.

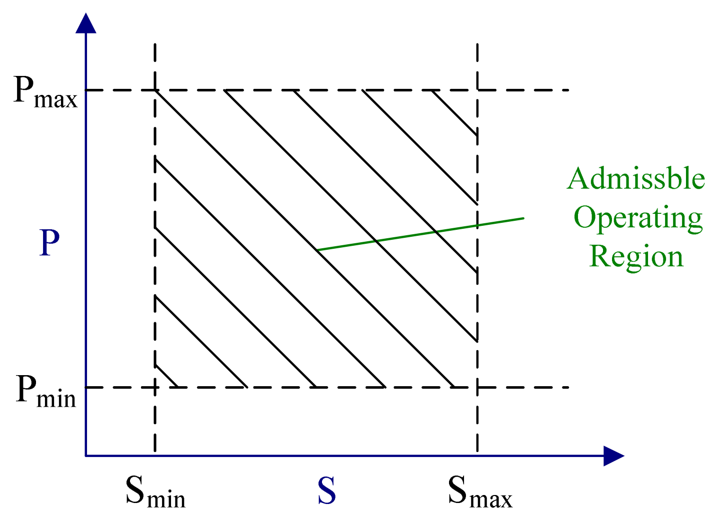

Figure 7.1 The conceptually admissible operating region of the model

The performance and security metrics allow us to quantitatively calculate how much protection a security mechanism can provide and how much performance will be degraded by using that security mechanism. Hence, we can make the trade-off decision between performance and security by adjusting the system performance and security parameters, within the boundary of the system requirements.

When both the performance and security requirements are satisfied, the system can use the available resources for improving performance and/or security. The performance-security trade-off model can be formulated as an optimization problem as follows:

$$Min\begin{cases} P = f(K_p, K_I, L_{sc}, L_{ca}) \\ S = g(K_p, K_I, L_{sc}, L_{ca}) \end{cases}, \tag{63}$$

with $P_{\min} < P < P_{\max}$, $S_{\min} < S < S_{\max}$.

With the trade-off objective functions, the system can compute and choose the best security parameters together with optimal performance parameters to optimize the overall system performance and security.

7.2 Optimizing Performance-Security Trade-off Based on CGA

This section presents a framework of performance-security trade-off optimization based on CGA.

7.2.1 Coevolutionary Genetic Algorithm

We selected the CGA for solving this trade-off optimization problem because it is robust for solving complex, non-linear system optimization problems, (e.g. NCS). CGA is noted by its fast convergence while maintaining a good diversity of solutions in EAs [86-88]. It is fairly immune to the local minima and non-linear nature of the problems being considered in this paper. Moreover, the CGA's performance is robust to the size of the problem, or the dimension limits. Thus, CGA is particularly suitable for solving this trade-off optimization problem when compared to other approaches, such as GA and Particle Swarm Optimization (PSO) [89].

Similar to genetic algorithms inspired by nature, the concept of coevolution (used as the foundation for CGA) comes from biological observations. Indeed, nature is composed of several species that coevolve, i.e., an individual's evolution is related to the evolution of other species. Instead of considering a population of similar individuals representing a global solution (like conventional genetic algorithms do), CGA considers the coevolution of subpopulations of individuals representing specific parts of the global solution.

The pseudo-code of the CGA is given in Table 7-1, in which the evolution of each species is handled by a standard GA.

Table 7-1 The Pseudo-code of CGA

Algorithm: CGA
Gen = 0
for each *species i* **do**
begin
Initialize the species population *Pop[i][0]*
Evaluate fitness of each individual *Ind[i][0][n]* in *Pop[i][0]*
Choose a representative *Rep[i][0]* from *Pop[i][0]*
end
while *termination condition = false* **do**
begin
for each *species i* **do**
begin
Reproduction from *Pop[i][Gen]* to get *Mate[i][Gen]*
Crossover and mutation from *Mate[i][Gen]* to get *Pop[i][Gen+1]*
Evaluate fitness of each individual *Ind[i][Gen+1][n]* in *Pop[i][Gen+1]*
begin
Form a collaborations with representatives from all the species *(Rep[1][Gen], Rep[2][Gen],…, Rep[i][Gen], …)*
Calculate the fitness value of each individual *Ind[i][Gen+1][n]* with all the species' representatives
end
Choose a representative *Rep[i][Gen+1]* from *Pop[i][Gen+1]*
end
Gen = Gen + 1
end

7.2.2 Performance-Security Trade-off Model based on CGA

The performance-security trade-off model of a networked DC motor system based on CGA is depicted in Figure 7.2. We use an agent-based simulation method to evaluate the performance-security trade-off, in which the two objectives in the system under investigation are represented by two agents in the model: performance agent and security agent. The agents interact with each other through the system model. Each agent submits its decision variables to the system model and makes its decisions based on the knowledge of itself and the environment. Here, the system model consists of the DC motor, networked controller, security mechanism, and communication channel dynamics. Performance and security agents submit their decision variables to the system model simultaneously. Performance agent submits the control parameters (K_P and K_I,) while security agent submits the encryption parameters (L_{sc} and L_{ca}). The system model then calculates the system response according to the system dynamics. Then each agent calculates its fitness function with the system response and the decision variables it submits. In this framework, each agent only needs to optimize its own objective (fitness value), with the control or encryption parameters as the decision variables. The definition of each agent is listed in Table 7-2.

The schematics of the CGA optimization process of the secured networked DC motor is shown in Figure 7.3. Each agent is represented by a species (a species means a population of GA in this algorithm) in the ecosystem in Figure 7.3. Each species evolves a bundle of individuals, which represent the candidate decision variables of the corresponding agent. Each species is evolved through the repeated application of a conventional GA. Figure 7.3

also shows the fitness evaluation phase of the GA. For example, to evaluate an individual (control parameters) from species 1 (performance agent), collaboration is formed with representatives (optimal encryption parameters) from the other species (security agent). The system model solves for the system variable (system time response). Then species 1 can use the system variable to evaluate the fitness of its individual. Here the fitness function is the metric of the corresponding agent. There are many possible methods for choosing representatives with which to collaborate. An obvious one is to simply let the current best individual from each species be the representative.

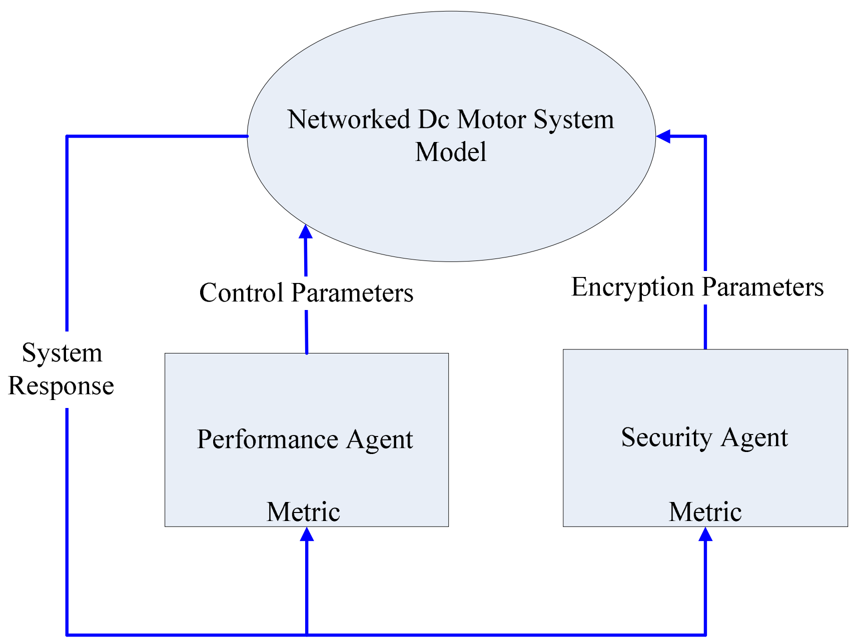

Figure 7.2 The performance-security trade-off model based on CGA

Table 7-2 The Definitions of Performance and Security Agents in The Trade-off Model

The Performance Agent:
Decision variables (Control parameters): K_p, K_I Objective (fitness function): min $P = f(K_p, K_I, L_{sc}, L_{ca})$ Constraints: $P_{\min} < P < P_{\max}$
The Security Agent:
Decision variables (Encryption parameters): L_{sc}, L_{ca} Objective (fitness function): min $S = g(K_p, K_I, L_{sc}, L_{ca})$ Constraints: $S_{\min} < S < S_{\max}$

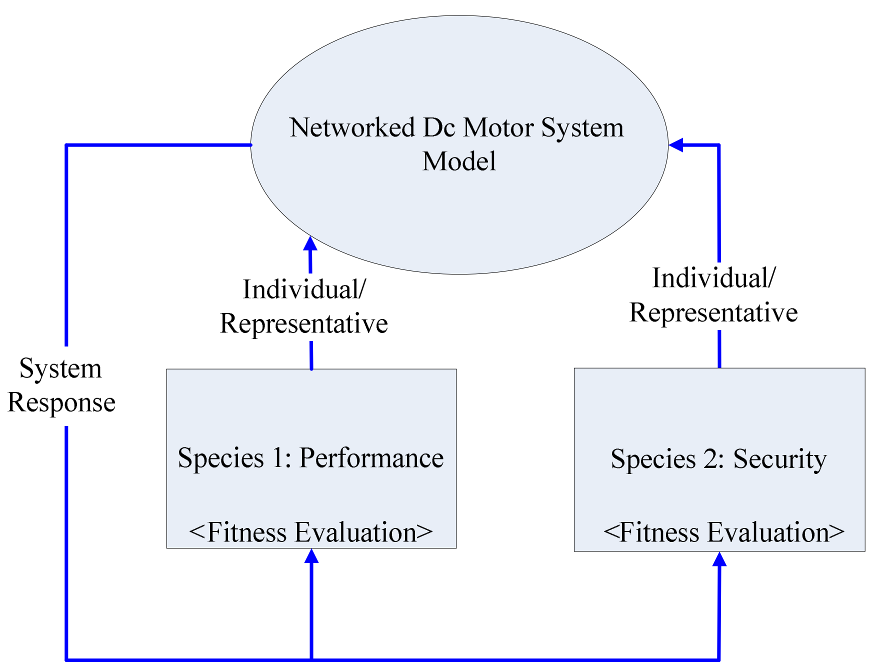

Figure 7.3 Framework of the CGA optimization process

In the CGA optimization process, in order to facilitate the fast convergence of the simulation, a "greedy" representative-selecting method is used in which the current best individual from each species is selected as the representative of that species [83]. The simple roulette wheel selection method is used in generating the reproduction operator of each species. From the evaluation process above, we can see that the species are coordinated by the system response. When one agent changes its decision variables to gain better fitness value, it will change the system response according to the system dynamics and, in turn, changes the fitness value of the other agent. The other agent will behave in the same way. The adjustment process will continue until no one can gain better fitness value by changing its own decision variables without changing the decision variables of the other agent. In other words, the performance-security trade-off reaches a Nash equilibrium.

7.2.3 Pareto Optimal Model based on CGA

An attractive feature of the Nash equilibrium as a solution concept is that, while being general, the set of individuals it represents can be relatively small; this is a valuable property for the coevolutionary search. However, a disadvantage is that part of the Nash equilibrium may be dominated. This means that finding a Nash equilibrium does not guarantee that the solution is optimal. Since the Nash equilibrium is generally Pareto-inefficient, the performance and security agents may improve their fitness by jointly modifying their decision variables. Thus, a set of Pareto optimal solutions – Pareto front – may exist. Since there are many Pareto optimal solutions in the Pareto front, a particular optimal solution is dependent on any additional conditions imposed on the system. This paper will only focus on searching for the general Pareto Front of the model.

Table 7-3 The Definitions of Performance and Security Agents in Pareto Optimal Model

The Performance Agent:
Decision variables (Control parameters): K_p, K_I
Objective (fitness function): $Min\begin{cases} P = f(K_p, K_I, L_{sc}, L_{ca}) \\ S = g(K_p, K_I, L_{sc}, L_{ca}) \end{cases}$
Constraints: $P_{\min} < P < P_{\max}$
The Security Agent:
Decision variables (Encryption parameters): L_{sc}, L_{ca}
Objective (fitness function.): $Min\begin{cases} P = f(K_p, K_I, L_{sc}, L_{ca}) \\ S = g(K_p, K_I, L_{sc}, L_{ca}) \end{cases}$
Constraints: $S_{\min} < S < S_{\max}$

The definitions of the performance and security agents in the Pareto optimal model are listed in Table 7-3. As shown in Table 7-3, the fitness functions of these two agents are the same since they are jointly improving their fitness in the Pareto optimal model. The Pareto optimal model actually forms a classical multi-objective optimization problem with inequality constraints. The same CGA optimizing process is used for solving this multi-objective optimization problem. Either the performance agent or the security agent should optimize both of their fitness values in the system, with its own parameters as the decision variables. Apparently, in a solution of this multi-objective optimization problem, no agent can achieve a better fitness value without the change of the other agent's fitness value. Thus, the ability of CGA to facilitate the interactions among the different agents provides a good platform of the modeling and simulation of the trade-off optimization problem.

7.3 Simulations and Experiments

In order to obtain the optimal performance and security parameters of the model and validate the algorithms described in this chapter, a Secured Networked DC Motor System Test-bed and a corresponding Networked DC Motor System Simulator were developed. In the test-bed, the DC motor system was integrated with a networked PI controller to control the speed of the motor over a network. Our experimental task is to design the control and security parameters for the system to drive the networked DC motor to a reference speed 50 rad/s while achieving the optimal performance and security values. And the experiment should follow the steps below:

- Step 1: Set up the actual Secured Networked DC Motor System Test-bed.
- Step 2: Model the actual system with the Networked DC Motor System Simulator in Matlab Simulink.
- Step 3: Use CGA to compute the optimal control and security parameters in the Simulator.
- Step 4: Apply the optimized parameters in the test-bed to validate the results.

7.3.1 Secured Networked DC Motor System Test-bed Setup

The Secured Networked DC Motor System Test-bed is composed of four parts:

- **The DC Motor Platform:** The DC motor platform used in this test-bed is the Feedback MS150 Modular Servo System. This platform is intended for experimental use in the education and research of closed-loop systems studies.
- **The Networked PI Controller:** The PI speed controller was implemented in Labview 8.2 (windows XP). All the control parameters and system parameters such as PI gains (K_P, K_I) and sampling period h are controllable by users via a Graphical User Interface (GUI) implemented in Labview.
- **The Security Simulator:** Different security algorithms were implemented on the security simulator developed in Labview to study the effect of different security mechanisms.
- **The Network Simulator:** In this test-bed, the PI controller in Labview and the DC motor platform was hard-wired via a National Instrument DAQ board 6009 instead of a wireless network to read and write analog system variable values to and from the DC motor platform to the digital controller in Labview. However, a wireless network environment was simulated by the network simulator developed using Labview and a microcontroller. By doing this, each wireless network delays of interest can be controlled so that the delay effects on the overall system can be studied.

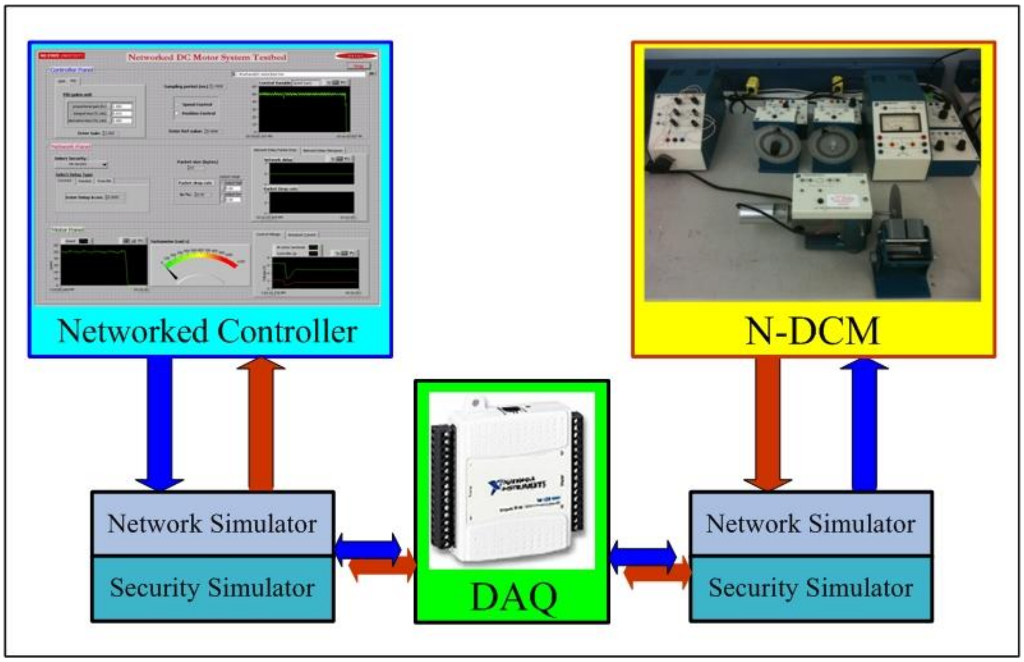

Figure 7.4 Secured Networked DC Motor System Test-bed

This test-bed allows us to design repeatable experiments under a wide range of network environments, system conditions, and network settings. The system structure of the entire test-bed is depicted in Figure 7.4.

7.3.2 Networked DC Motor System Simulator Setup

As shown in Figure 7.5, similar to Section 6.3, a corresponding simulator is implemented in Matlab Simulink according to the testbed setup and system description in Section 7.1. The DC motor dynamics is simulated using differential equations. A wireless network environment with random time delay and packet drop rate was simulated such that each desired network parameter can be user controlled for the study of parameter effects on the network controller. The security mechanism is simulated as a discrete event system by the State Flow in the Simulink. Several network security algorithms have been implemented to simulate the effect of network security additions.

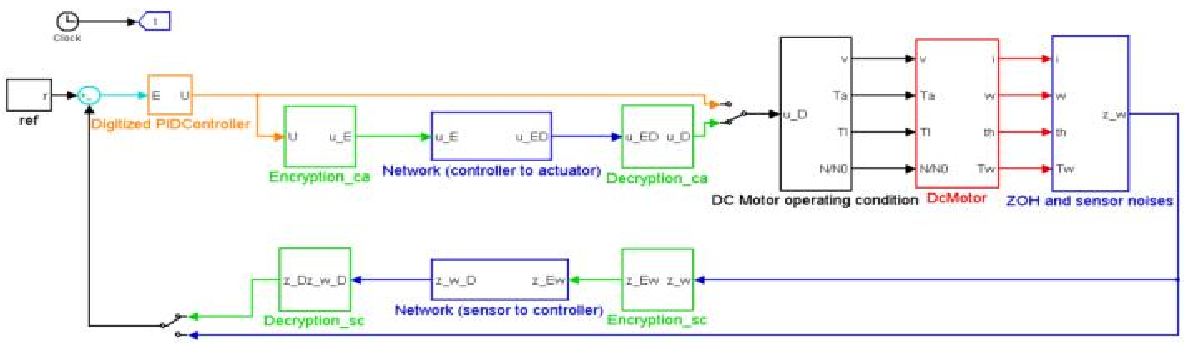

Figure 7.5 Networked DC Motor System Simulator based on Matlab Simulink

The parameters of the DC motor used in this chapter are the same as those listed in Table 6-2. The parameters are obtained via system identification techniques on the DC motor in the test-bed. The sensor and the output data are sampled with sampling period h = 1 ms by considering the buffer size = 1, which means that the measured data will be immediately used by the PI controller.

The network transmission delay on the Internet is random in nature. Generally speaking, it is in the order of a few hundred of a millisecond, typically 100 ms to 600 ms on the Internet. Therefore, τ_{sc}^{tran} and τ_{ca}^{tran} are set to be in the range of [100ms, 600ms]. The additional time delay imposed by security additions, τ_{sc}^{sec} and τ_{ca}^{sec}, are simulated according to the statistical results of the processing delay of AES encryption algorithms with different encryption key length on a Linux Machine (Kernel 2.4.18-3 i386). The performance metric specifications for the networked DC motor speed control system are: $\bar{E}^*$= 0.05. The weighting factors α = 380 and β = 1 are chosen using heuristics to normalize the security metric, thus the vulnerability value of AES is normalized to the same scale of the system oscillation level of the networked DC motor system. The CGA is coded in Matlab based on the GA Toolbox. The key variables and parameters of the CGA are defined in Table 7-4.

Table 7-4 The Key Variables and Parameters of CGA

Individual	A set of decision variables in one species.
Population	All the individuals in one species. Individuals of each species are randomly initialized within the constraints.
Generation	Number of iterations in the evolutionary process.
Reproduction	Simple roulette wheel selection.
Crossover	Two-point crossover operator: two crossover positions are selected, and the two parents swap the strings between them with certain probability.
Mutation	Bit-wise mutation operator: each bit of the individual is randomly changing from 1 to 0 or vice versa with certain probability.
Elitist Strategy	Preserving Elite Rate of the best individuals in former generation.
Representative Choice	Current best individual from each species.
Termination Condition	The number of generations reaches its preset upper limit.

7.3.3 Simulation Results: Trade-off model

To compute the optimal parameters, we apply the CGA optimization to several cases in order to draw statistically significant results. The CGA parameters of the test cases are listed in Table 7-5.

For each case, we run the simulation repeatedly 50 times to get the average results. The simulation results are listed in Table 7-6. Although they are on different convergence curves, all the cases reach the Nash equilibrium of this problem, which is (2.32, 52.52) with the corresponding decision variables ($K_P = 0.23$, $K_I = 0.69$, $L_{sc} = 190$ and $L_{ca} = 190$).

Table 7-5 Test Cases' CGA Parameters

Test case #	*population size*	*generations*	*mutation rate*	*crossover rate*	*elite rate*
1	50	200	0.4	0.4	0.2
2	50	200	0.4	0.5	0.1
3	50	200	0.3	0.6	0.1
4	50	200	0.2	0.7	0.1

Table 7-6 Nash Equilibrium Results of Different Cases

Test Case #	Performance Value		Security Value	
	Mean	95% CI	Mean	95% CI
1	2.3198	[2.3198,2.3199]	52.5186	[52.5186,52.5187]
2	2.3200	[2.3198,2.3201]	52.5188	[52.5186,52.5189]
3	2.3199	[2.3197,2.3201]	52.5187	[52.5185,52.5189]
4	2.3201	[2.3196,2.3205]	52.5189	[52.5184,52.5193]

The variations of performance and security fitness values with respect to the number of generations in the typical runs of the simulation for each case are shown in Figure 7.6 and Figure 7.7, respectively. As we can see, the fitness values in all cases converged to the Nash equilibrium rapidly after certain generations of coevolution. Thus, CGA has high efficiency in the trade-off optimization problem.

From the simulation results, we can observe and conclude that CGA can find the Nash equilibrium in all cases for this problem, which demonstrates that the proposed CGA framework provides a satisfactory modeling and optimization for the performance-security trade-off on a networked DC motor system.

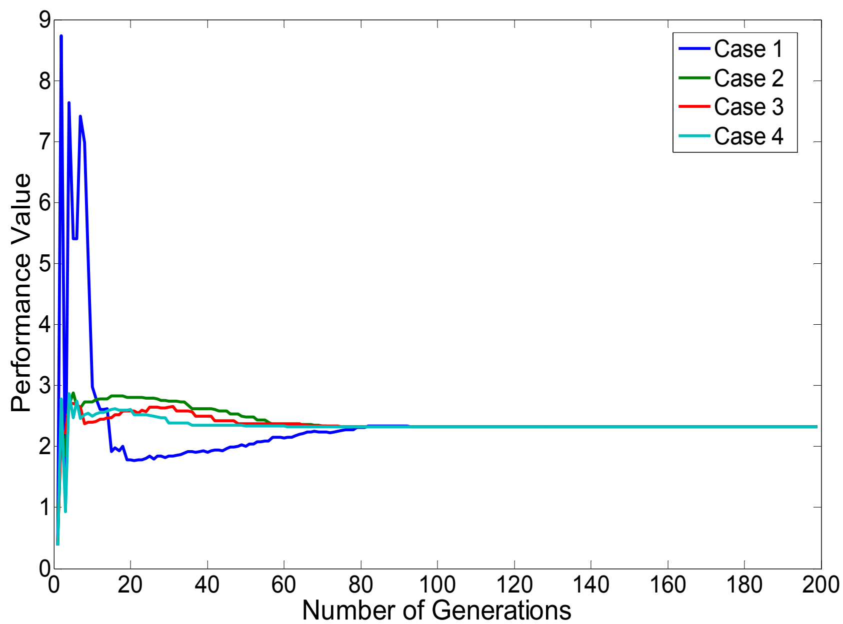

Figure 7.6 Variations of performance fitness values in CGA process

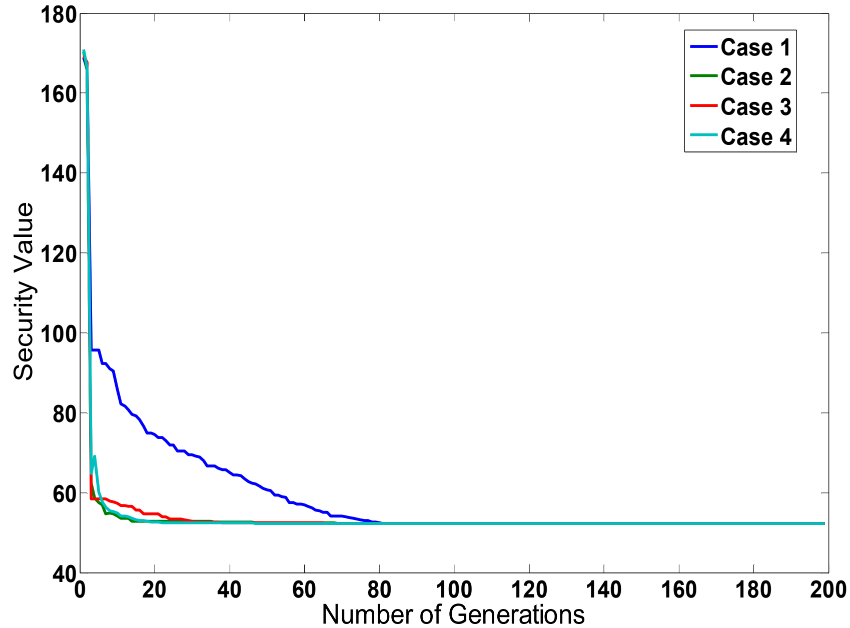

Figure 7.7 Variations of security fitness values in CGA process

7.3.4 Simulation Results: Pareto Optimal Model

To validate the Nash equilibrium as the optimal solution, an additional Pareto optimal model simulation is run as well. The simulation parameters are the same as those in the trade-off model simulations above. The Pareto front obtained from the CGA optimization is plotted in Figure 7.8. Since the Pareto optimal solutions are not unique, we can select the particular optimal solution dependent on whatever additional conditions are imposed on the system. However, we find that the Nash equilibrium we obtained from the trade-off model based on CGA lies on the Pareto front in this case. Thus, the Nash equilibrium obtained is Pareto-efficient, which guarantees that it is an optimal solution of this problem.

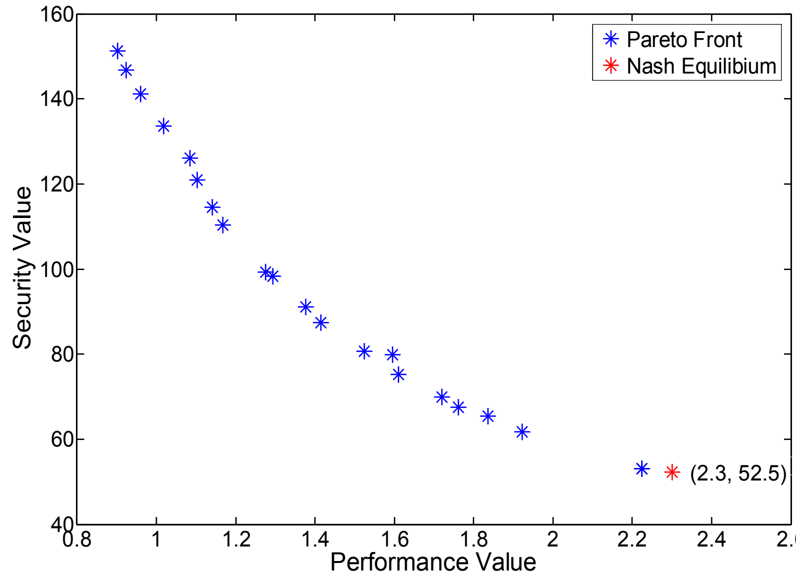

Figure 7.8 Pareto front of the Pareto optimal model based on CGA

7.3.5 Simulation Results: Comparison

We also compared the optimization performance of the CGA with standard GA and PSO [90]. A paired t-test was conducted to compare the mean optimal performance and security values of different optimization methods under the same setting (population size = 50, generations = 200).

As we can see in Table 7-7, all the confidence levels do not cover zero. All of these conditions conclude that there is a statistically significant difference between the performance and security values for each pair of compared algorithms at 5% level of significance. Table 7-8 shows the averaged amount of increase in the optimal minimum performance and security values of other optimization algorithms as compared to the CGA. This value was computed as:

$$\%P\_Increase = \frac{P_{GA/PSO} - P_{CGA}}{P_{CGA}} \times 100\%, \tag{64}$$

$$\%S\_Increase = \frac{S_{GA/PSO} - S_{CGA}}{S_{CGA}} \times 100\%. \tag{65}$$

Table 7-8 suggests that the increases in optimal minimum performance and security values are practically significant when using GA and PSO rather than CGA. This means the CGA outperforms GA and PSO in finding the optimal values in this performance-security trade-off optimization problem on NCS.

Table 7-7 Paired T-Test Comparing The Mean Optimal Performance and Security Values of Different Optimization Algorithms

Pair of Optimization Algorithms	95% CI of Security Value Difference	95% CI of Performance Value Difference
(CGA, GA)	[-0.0048, -0.0026]	[-0.0416, -0.0155]
(CGA, PSO)	[-0.0173, -0.0082]	[-0.1136, -0.0923]

Table 7-8 Percentage Increase in Optimal Minimum Performance and Security Values using Other Optimization Algorithms

Optimization Algorithm	GA	PSO
% Increase in Performance Value	3.15%	6.54%
% Increase in Security Value	1.48%	2.55%

7.3.6 Experiment Results:

After obtaining the design parameter values of K_P, K_I, L_{sc} and L_{ca} from the simulator, we then implemented these values on the actual Networked DC Motor System Test-bed to validate the results.

Table 7-9 Averaged Results of Contrast Experiments

Parameters Setting	Performance Value	Optimal Setting Improvement	Security Value	Optimal Setting Improvement
Nominal	5.6323	56.67%	61.4543	14.34%
Stability Edge	610.6299	99.60%	660.8290	92.03%
Minimum Operating	12.5980	80.63%	75.9313	30.67%
Optimal	2.4406	-	52.6394	-

Table 7-10 The Result of Step Responses for Different Settings

Parameters Setting	2% Setting Time (s)	Overshoot	Rising Time (s)
Nominal	0.57	82.2%	0.055
Optimal	0.2	9.5%	0.078

The system setting and configurations of the test-bed are the same as the simulator. And the experimental task is still to drive the networked DC motor to a reference speed 50 rad/s.

To compare and evaluate the optimal performance and security parameters ($K_P = 0.23$, $K_I = 0.69$, $L_{sc} = 190$ and $L_{ca} = 190$), we run several experiments with different parameter settings, including the nominal parameters setting ($K_P = 1$, $K_I = 0.1$, $L_{sc} = 112$ and $L_{ca} = 112$), the stability-edge parameters setting ($K_P = 5$, $K_I = 1$, $L_{sc} = 190$ and $L_{ca} = 190$) and the minimum-operating parameters setting ($K_P = 0.1$, $K_I = 0.1$, $L_{sc} = 64$ and $L_{ca} = 64$).

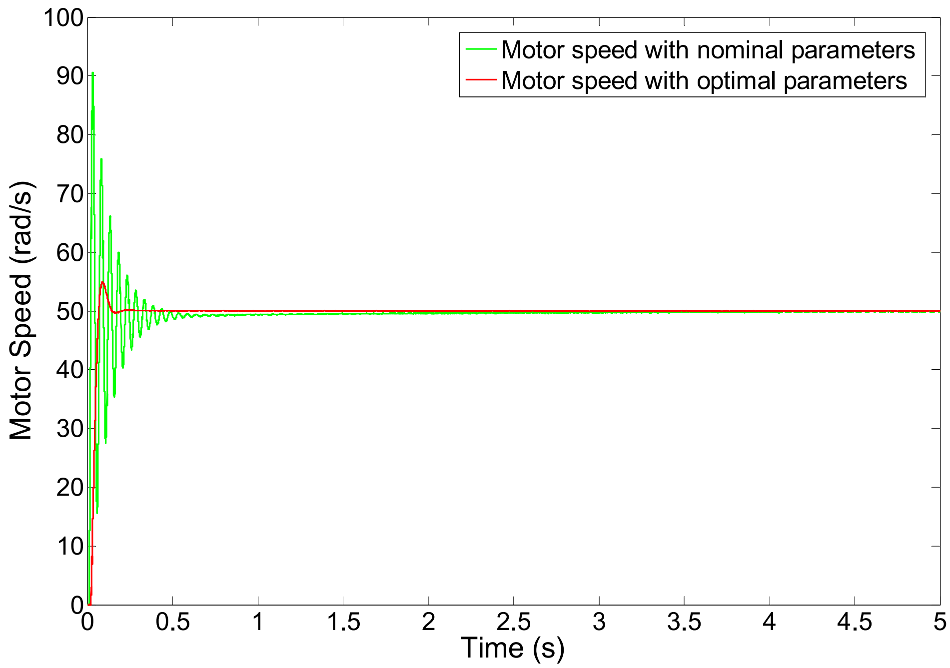

Figure 7.9 The step responses of the test-bed for different settings

The experimental results are listed in Table 7-9. For each setting, we run the experiments repeatedly 10 times to get the average results. Although all of these results are stable, the setting with the optimal parameters has an impressive improvement in both performance and security measures compared to the other parameter settings. Figure 7.9 shows the step responses of nominal and optimal settings from one experiment. The detailed result is listed in Table 7-10. We can see that the system with the optimal parameters has a much better system response than the nominal setting.

CHAPTER 8. PERFORMANCE AND SECURITY TRADE-OFF OPTIMIZATION IN D-NCS

With the rapid advancements in the Internet, embedded systems, and wireless communications technologies in recent years, research on D-NCS has been gaining popularity because of their high potential for widespread applications. Today, D-NCS have become even easier to implement with the tremendous increase in the deployments of wireless systems [91, 92], which have been used in a wide variety of applications, such as monitoring and operations for manufacturing plants, space projects, defense systems, robot navigations, nursing homes, traffic management, and many more.

To reduce operational costs and improve performance, D-NCS were transitioned to less expensive standardized technologies, operating systems, and protocols that are currently prevalent (e.g., on the Internet). As a result, real-time monitoring and control information is readily and easily accessible to a large number of people on the Internet. This also increases the vulnerability of the D-NCS to malicious network attacks. Thus, data sharing and communications security are among the main concerns in D-NCS, considering their time-sensitive and data-sensitive applications. It is critical to protect transmitted data from unauthorized access and modification in the D-NCS's communications channels.

Although D-NCS with security, from a control system's perspective, are still in their infancy, many D-NCS have been well protected by different levels of security mechanisms [93, 94]. However, adding more security may adversely affect system performance because of limited system resources. The impact of the security mechanisms on system performance has not been addressed thoroughly. Security requirements are often in competition with performance requirements, such as real-time dynamic performance that is limited by system resources [95] and the extra time delay imposed by additional security. Thus, there is a trade-off between the D-NCS's performance and its security measures.

Since the traditional D-NCS without security protection is vulnerable to various security attacks [96], developing security mechanisms for D-NCS has turned out to be a research hotspot, giving rise to many topics, such as performance assessments for D-NCS with security, novel network architectures to support security for D-NCS, and novel intrusion detection schemes for D-NCS [97, 98].

Most of recent efforts deal with the security issues of the D-NCS from the prevention and detection perspectives. However, there is also a growing demand for studying the impact of the security additions on the systems' performance and the trade-off between them. Gupta *et al.* [64] characterized the D-NCS application on the basis of the security's effect on its performance and mapped the added security features to an increased time delay in the system to show this trade-off for a path-tracking application. A trade-off model for conventional networked control systems has been described in Chapter 6 [37]. A set of quantitative performance and security metrics have also been developed and combined in a tradeoff objective function. A successful two-agent case of optimizing the trade-off between the system dynamic performance and on-demand security on a networked DC Motor system using CGA has also been demonstrated in Chapter 7 [99]. However, D-NCS often have more than two agents. Most of these systems are multi-agent systems composed of several distributed sensors, controllers, and actuators, and all of these components are connected over a network. Thus, in this chaptetr, we use one kind of D-NCS—the Intelligent Space (iSpace)—as an example of how to analyze and optimize the trade-off between system dynamic performance and on-demand security. A quantitative performance-security trade-off model for D-NCS using iSpace as an illustration is presented and a paradigm of performance and security trade-off optimization is proposed based on CGA. The proposed trade-off model can be used to simultaneously adjust system resources to provide sufficient protection and satisfy real-time performance requirements for D-NCS, while the CGA is an effective approach for this trade-off analysis and optimization by its fast convergence while maintaining a good diversity of solutions in EAs and immunity to the local minima and non-linear nature of the optimization problems being considered in D-NCS.

8.1 iSpace System Model

iSpace system at ADAC Lab has already been partially introduced in Chapter 4. It is a network-based integrated navigation system with different modules combined together to guide several unmanned ground vehicles (UGVs) from one point to another where the navigational intelligence comes from remote controllers. It includes a network of distributed sensors (cameras and encoders) and multiple actuators (UGVs). iSpace aggregates the status of the entire space from each agent's sensory information and responds intelligently to the system's goals. iSpace system has several major components: distributed sensors and actuators, hybrid hierarchical and distributed controllers, the system's security manager, and the communication network. The system structure has been shown in Figure 4.3.

The D-NCS in this chapter is composed of multiple UGVs controlled by iSpace that move between locations tracking a path that avoids obstacles in a secured network environment. Thus, UGVs need to track the path and reach their destination efficiently while the security manager has to protect the whole system from malicious network attacks. However, the system resources (network bandwidth) are limited, so the UGVs and system security manager are actually competing for the same resources to achieve their goals.

8.1.1 UGV Dynamics

The UGV dynamics has been described in previous chapters. See Section 4.3.1.

8.1.2 Path-Tracking Controller

The quadratic curve (QC) path-tracking controller defined in [100], the feedback preprocessor (FP), and the predictive control gain scheduling middleware (PCGSM) [101, 102] are used to control the UGVs. The path-tracking controller calculates a path, $\boldsymbol{p}$, defined by a set of consecutive waypoints on the ground for each UGV to follow as $\boldsymbol{p} = [\boldsymbol{p_x} \;\; \boldsymbol{p_y}]$, where $\boldsymbol{p_x}$ and $\boldsymbol{p_y}$ are sets of the world frame x and y coordinates, respectively. The QC path-tracking algorithm will determine a look-ahead distance based on previous UGV control. A reference point ahead of the UGV is constrained to move along $\boldsymbol{p}$. The location of the reference point is determined by finding the nearest point on the path to the UGV and looking ahead on the path a variable distance defined by using the methods in [100].

To track the reference point, the path-tracking controller finds a QC that passes through the reference point and has its vertex located at the UGV. Since the sampling period is sufficiently small, reference velocity $\boldsymbol{v}_{ref}$ and turn rate $\boldsymbol{\omega}_{ref}$ can be calculated by fitting a circle in the QC [100] as:

$$A = \frac{e_y}{e_x^2}, \tag{66}$$

$$v_{ref} = \mathrm{sgn}(e_x)\frac{\alpha}{1+|A|}, \tag{67}$$

$$w_{ref} = 2Av_{ref}, \tag{68}$$

where A is the QC coefficient and α is the maximum velocity. Using A to determine the velocity causes the UGV to slow down while turning to prevent error. Since the reference command is a circular trajectory defined by (67) and (68), the path-tracking controller needs to frequently generate new QCs and subsequent circular arcs for the UGV. The UGV will follow the reference's circular trajectory for a short time so that when all of the circular arcs are aggregated together, they will approximate the QC, which in turn approximates the desired path.

8.1.3 Security Manager

To protect iSpace from malicious network attack, this paper focuses on the confidentiality aspect of the security service using secret key cryptography. The secret key algorithm of the Advanced Encryption Standard (AES) is a symmetric cipher considered secure for wireless systems. It works with 128 bits of block size with key sizes of 128 or more. As Electronic Code Book (ECB) is considered to be the fastest mode of operations, it is used frequently in real-time systems. Thus, in this paper, the AES with an ECB mode is integrated with the application to encrypt and decrypt the information flow along the communication network [75]. Thus, the system must provide the necessary bandwidth for the security mechanisms in the communication channels.

8.1.4 Bandwidth Allocation

For simplicity, the bandwidth allocation algorithm used in this paper allocates the bandwidth according to the request of each UGV proportionally.

Each UGV should have a feasible safety region along the path. Outside of this region, the UGV may oscillate, out of control and collide with other UGVs. Therefore, UGVs must remain within the safety region to maintain stability. The stability criterion is then defined as:

$$\max\left(e_i(t)\right) \leq \delta_i(t), \tag{69}$$

where $\delta_i(t)$ is the safety region around the path and $e_i(t)$ is the distance from the UGV coordinates to the nearest path point for the ith UGV. By putting the system dynamics and the error formulation in the optimization problem, the maximum sampling time $\Delta T_{i,max}$ that guarantees the stability of the ith UGV is then calculated by solving the algebraic equation:

$$e_i(t+\Delta T_{i,\max})=\delta_i(t). \tag{70}$$

Thus, the minimum bandwidth required in bits per second (bps) that guarantees stability (69) is:

$$BW_{i,\min}=\frac{L\times 8}{\Delta T_{i,\max}}, \tag{71}$$

where L is the size of the original data packet in bytes. The bandwidth needed by the necessary security mechanism is:

$$BW_{\sec}=\sum_{i=1}^{n}\frac{L_{\sec}\times 8}{\Delta T_i}, \tag{72}$$

where L_{sec} is the size of the security addition to the encrypted packet in bytes, which depends on the encryption algorithms. ΔT_i is the actual sampling time of the ith UGV.

So, the stability of every loop in the system is guaranteed as long as (73) is satisfied:

$$\sum_{i=1}^{n}BW_{i,\min}+BW_{\sec}\le BW_{Total}, \tag{73}$$

where n is the number of control loops (UGVs). The remaining bandwidth is shared among the control loops (UGVs) by proportionally allocating it according to the request of each UGV, $BW_{i,req}$. The actual bandwidth allocated to each UGV is calculated as:

$$BW_i = BW_{i,\min} + \left(\frac{BW_{i,req}}{\sum_{i=1}^{n} BW_{i,req}} \right) \left(BW_{Total} - \sum_{i=1}^{n} BW_{i,\min} - BW_{\sec} \right). \tag{74}$$

8.2 Trade-off Model Formulation

This section describes the performance-security trade-off model of the iSpace system, including the quantitative performance and security metrics and the tradeoff objective function.

8.2.1 The Performance Metric

To measure the performance of the UGVs in iSpace, the performance metric, P_i, has been implemented. P_i is the accumulated error calculated as the area between the desired path and the actual path of the UGVs:

$$P_i = \int_{t_{i,0}}^{t_{i,f}} (a_i(t) - r_i(t))^2 \, dt, \; i = 1,2,\ldots,n, \tag{75}$$

where $a_i(t)$ is the actual path that the ith UGV travels, and $r_i(t)$ is the desired path that the ith UGV should track.

8.2.2 The Security Metric

Existing qualitative metrics classify various security mechanisms to several discrete levels, such as low, medium, and high. Security mechanisms in the higher level can provide better protection than those in the lower levels. But it's impossible to compare security mechanisms within the same security level. Furthermore, qualitative metrics are too coarse for fine control of the trade-off between dynamic performance and security in D-NCS. Thus, a quantitative metric that generates a security strength value for each security mechanism is used in this paper and, hence, is more suitable for a quantitative comparison of the security strength of any two security mechanisms.

Without considering any shortcut attacks, brute force attack is the only way used to crack the encryption key in this paper. For example, an AES cipher with a key length of 128 bits has 2^{128} possible key combinations. Assuming unit complexity for testing one key, the worst-case complexity involved in cracking this 128-bit AES cipher is 2^{128}.

With this assumption, the security level of an encrypted message frame is decided by its encryption key length. Thus, a security measure with respect to brute force attacks is described as $S_{BF}(N) = log_2(N)$, where N is the encryption key length. The security metric of the system is then measured by the vulnerability of its encryption algorithms to brute force attackers, as:

$$S = \frac{nS_{\max}}{\sum_{i=1}^{n} \log_2 N_i}, \; i = 1,2,\ldots,n, \tag{76}$$

where $S_{\max} = \max(\log_2 N_i)$.

8.2.3 The Trade-off Objective Function

With the defined performance metric, P_i, and the security metric, S, the operational requirements can be formulated quantitatively. Thus, we need to add the following constraints:

$$P_{i,l} < P_i < P_{i,u}\,,\ S_l < S < S_u\,, \tag{77}$$

where $P_{i,l}$ and $P_{i,u}$ are the operational lower and upper bounds, respectively, of dynamic performance for the ith UGV, S_l and S_u are the operational lower and upper bounds, respectively, of the system security level.

The performance and security metrics allow us to quantitatively calculate how much protection a security mechanism can provide and how much performance will be degraded if we use it. Therefore, we can make the trade-off decision between performance and security by adjusting the system resource—allocated bandwidth—within the boundaries of the system operational requirements.

When all the performance and security requirements are satisfied, the system can use the remaining available resources to improve performance, security, or both. A trade-off objective function among the performance metrics and the security metric is formulated as the utility function below:

$$Min\ U = w_1 P_1 + w_2 P_2 + \ldots + w_n P_n + w_{n+1} S \tag{78}$$

subject to: $P_{i,l} < P_i < P_{i,u}\,,\ S_l < S < S_u\,,\ i = 1,2,\ldots,n\,.$

where w_j $(j=1,2,\ldots,n+1)$ are the weighting factors representing the preferences on different metrics. With different applications, the weighting factors can be adjusted accordingly. In this paper, all the factors are equally weighted.

With this trade-off objective function, the system can compute and choose the best allocated bandwidth for the security manager together with the optimal allocated bandwidth for each UGV in order to optimize the overall system performance and security.

8.3 Trade-off Optimization Using the CGA

This section presents the paradigm of the performance and security trade-off optimization for iSpace based on the CGA.

8.3.1 Coevolutionary Genetic Algorithm

Similar to genetic algorithms inspired by nature, the concept of coevolution—used as the foundation for the CGA—comes from biological observations. Nature is composed of several species that coevolve. Instead of considering a population of similar individuals that represent a global solution as conventional genetic algorithms (GAs) do, the CGA ponders the coevolution of subpopulations of individuals representing specific parts of the global solution [88].

The pseudo-code of the CGA has been shown in Table 7-1, in which the evolution of each species is handled by a standard GA, while the evaluation of an individual from each species is handled through collaboration with representatives from other species.

8.3.2 Performance-Security Trade-off Model Based on CGA

The performance-security trade-off model of D-NCS based on the CGA is depicted in Figure 8.1. Since all the objectives in the trade-off objective function (78) are equally weighted, we use an agent-based simulation method to evaluate this trade-off problem. In this simulation all the objectives in the system under investigation are represented by the agents in the model: the performance agents and the security agent.

The definition of each agent (including its decision variables, fitness function and constraints) is shown in Table 8-1. As shown in Figure 8.1, all the agents interact with each other through the environment and form a non-cooperative game [103]. Each agent submits its decision variables ($BW_{i,req}$ *or* BW_{sec}) to the system environment model and takes its following actions based on the knowledge of itself and the system response BW_i from the environment. Here, the system environment consists of the total bandwidth BW_{Total} the system has. All the agents submit their decision variables to the system environment model at the same time. The system environment model calculates the response BW_i according to the environment dynamics using equation (74). Then, each agent can calculate its fitness function (P_i or S) with the system response BW_i by using equations (75) and (76). In this trade-off model, the performance agents and the security agent compete against each other for bandwidth to achieve their goals—minimize their own fitness functions, where a non-cooperative game is formulated. Together, these goals comprise the final trade-off objective of the whole system.

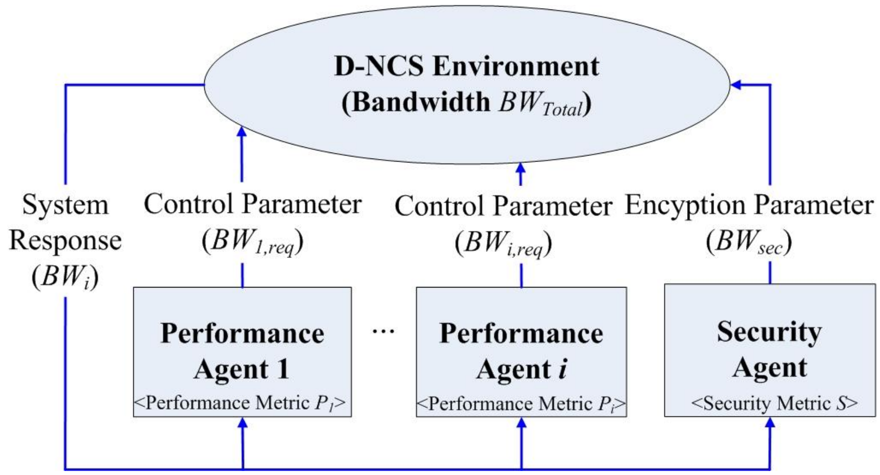

Figure 8.1 The performance-security trade-off model of D-NCS based on the CGA

Table 8-1 The Definition of Performance and Security Agents in the Trade-Off Model

The performance agent *i* (*i* = *1, 2 , ... ,n*):
Decision variables (Control parameters): $BW_{i,req}$
Objective (fitness function): min P_i
Constraints: $P_{i,l} < P_i < P_{i,u}$
The security agent:
Decision variables (Encryption parameters): $BW_{\sec}$
Objective (fitness function): min S
Constraints: $S_l < S < S_u$

The schematics of the CGA optimization process of D-NCS is shown in Figure 8.2. Each agent is represented by a species (a species means a population of GA in this algorithm) in the ecosystem shown in Figure 8.2 Each species evolves a bundle of individuals that represent the candidate competing strategies—decision variables of the corresponding agent. Each species is evolved through the repeated application of a conventional GA. Figure 8.2 also shows the fitness evaluation phase of the GA. For example, to evaluate an individual from the security agent, that individual must collaborate with representatives from the other species (performance agents). Then, the system environment model solves for the system response. The security species can use the system response variable to evaluate the fitness of its individual. Here, the fitness function is the metric of the corresponding agent.

For the representative selection, there are many possible methods for choosing the representatives with which to collaborate. In order to facilitate the fast convergence of the evolutionary process, we use a "greedy" method for selecting representatives. In this method the current best individual from each species is selected as the representative of that species. A simple roulette wheel selection method is used to generate the reproduction operator of each species.

From the evaluation process above, we can see that the species are coordinated by the system environment response. When one agent changes its decision variables to gain a better fitness value, it will change the system response according to system dynamics and, in turn, change the fitness values of the other agents. Other agents will behave in the same way. The adjustment process will continue until no agent can gain better fitness value by changing its own decision variables without changes of the decision variables of other agents. In other words, the trade-off finally reaches an optimal Nash equilibrium [103].

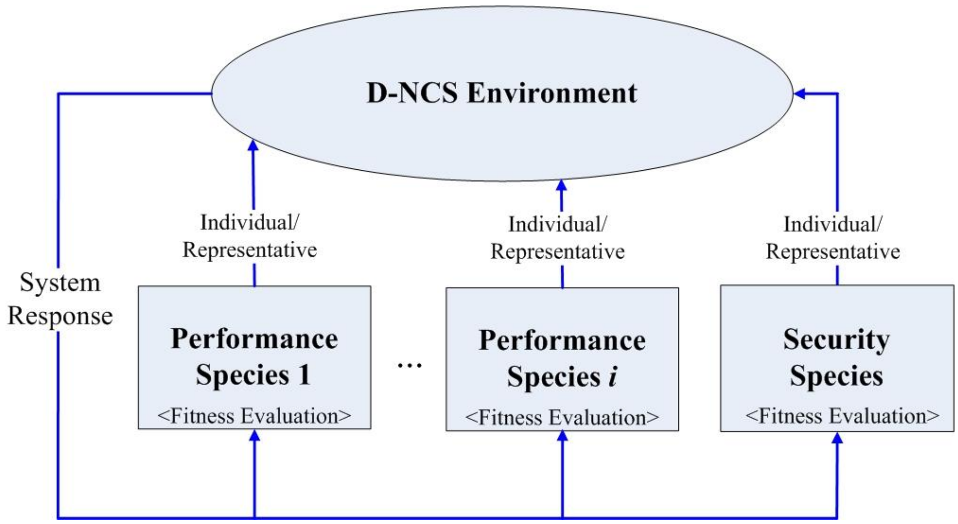

Figure 8.2 Framework of the CGA optimization process

8.4 Simulations

To obtain the system's optimal performance and security trade-off and validate the algorithms described in this chapter, we used the Simulink-based test-bed—iSpace Simulator—to evaluate the trade-off model in iSpace in real-time with utility constraints and varying inputs.

8.4.1 Test-Bed Description

The iSpace simulator that has been introduced and used in Chapter 4, is made up of the following parts:

- **Lego Mobile Robot:** The differential drive mobile robot used in this platform has been constructed out of modular off-the-shelf LEGO Mindstorms NXT, which provides an easy-to-use set of pieces and convenient motors with encoders. The robot has a wheel radius of 0.0286 meters (m) and a distance between wheels of 0.095m. With the gearing of the NXT motors, the maximum speed for this UGV is approximately 0.4 meters per second (m/s).

- **Networked Supervisory Controller:** The supervisory controller is used to combine the distributed sensor information and make control decisions to accomplish the system goal. It is implemented in a host PC. In addition to implementing the data fusion and connecting distributed sensors, the supervisory controller also makes control decisions (e.g., the QC path-tracking algorithm).
- **Security Simulator:** Different security algorithms (e.g., DES, AES) were implemented on the security simulator developed in Labview in order to study the effects of various security mechanisms. AES is only considered in this paper. Thus, $N_i \in \{128, 192, 256\}$.
- **Network Communication Simulator:** The main control and feedback information flow between the robot and the base station is carried over Bluetooth with supplementary sensing data (e.g., coordinates of the UGVs) coming from the image processing server based on the cameras' observations.
- **Graphical User Interface (GUI):** To provide a user interface for the system, the GUI communicates directly to the base station (supervisory controller). Through the GUI, the user can control connections to the UGVs. The user can issue manual commands to each of these connections and display the data acquired from them as

well. The progress and status of the UGVs are updated via the GUI while the UGVs are tracking the paths. Finally, the GUI also enables manual control and tuning of the UGVs through a command interface.

The test-bed lets us design repeatable experiments under a wide range of network environments, system conditions, and network settings. The system structure of the entire test-bed is depicted in Figure 4.5.

8.4.2 Simulation Conditions

Four UGVs are used in the simulation, so four test paths as shown in Figure 8.3 are used for the four UGVs that represent them in order to test the implemented method. The UGVs will be commanded to follow test paths using average bandwidth allocation and CGA-based bandwidth allocation with varying network delays, as characterized in Figure 8.3. We compare the effects of roundtrip time delays of 60 milliseconds (ms), 200ms, 400ms, and 600ms.

In order to facilitate testing, a software-based delay generator was also implemented. This delay generator acts on control signals and feedback to replicate the effects of network delay. To characterize the delay in this system, the UGV was pinged repeatedly with different delay settings for the delay generator. With the delay generator turned to 200ms, it will add 200ms of delay to each direction of communications. The results shown in Figure 8.4 indicate that when the delay generator is turned on at 60ms, 200ms, 400ms, and 600ms, the mean delays were 73ms, 237ms, 445ms, and 647ms, respectively.

The key variables and parameters of the CGA have been defined in Table 7-4 in Chapter 7. The operational bounds for each agent are shown in Table 8-2.

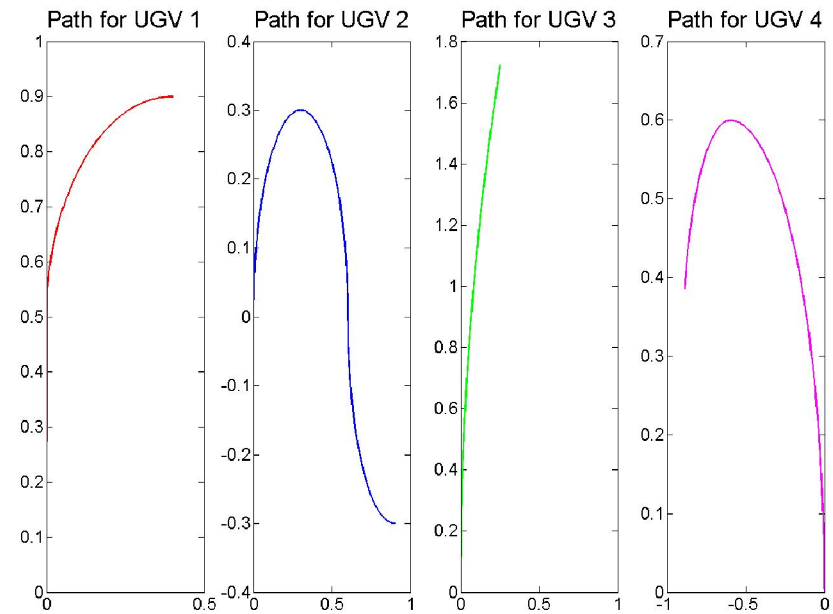

Figure 8.3 Test paths for the simulations

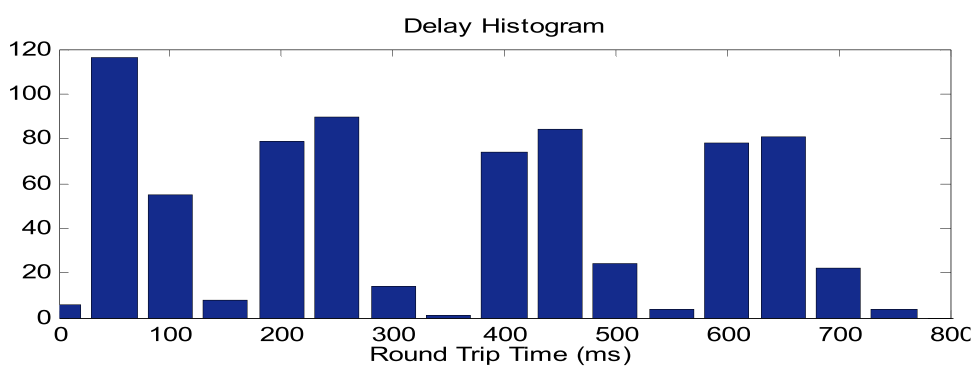

Figure 8.4 Network delay histogram for delay generator RTT settings of 60ms, 200ms, 400ms, and 600ms

Table 8-2 The Operational Bounds of Agents

Performance Agent	Lower Bound	Upper Bound	Security Agent	Lower Bound	Upper Bound
P_1	*0.01*	*0.25*	**S**	*1*	*1.15*
P_2	*0.03*	*0.50*			
P_3	*0.01*	*0.20*			
P_4	*0.02*	*0.50*			

8.4.3 Simulation Results: Different CGA Parameters

To compute the optimal allocated bandwidth for each agent, we apply CGA optimization to four cases in order to draw statistically significant results. The CGA parameters for the four test cases are listed in Table 8-3. The time delay is 60ms.

For each case, we run the simulation repeatedly 50 times to get the average results. The simulation results are listed in Table 8-4. Although they are on different convergence curves, all cases reach this problem's Nash equilibrium, which is (0.026, 0.067, 0.016, 0.043, 1.06) with the corresponding decision variables (BW_1=20.68%, BW_2 =55.46%, BW_3=3.76%, BW_4=14.1%, BW_{sec}=6%).

As we can see from the results in Table 8-4, the fitness values in all cases converged to the Nash equilibrium rapidly after a certain number of generations of coevolution. Thus, the CGA has a relatively high efficiency in the trade-off optimization problem. Figure 8.5 shows the tracking result of all the UGVs from test case 2. We can see that the system with the optimal bandwidth allocation has a very good path-tracking performance for all four UGVs.

Table 8-3 The CGA Parameters of the Four Test Cases

Test case #	population size	generations	mutation rate	crossover rate	elite rate
1	*100*	*500*	*0.4*	*0.5*	*0.1*
2	*100*	*500*	*0.3*	*0.6*	*0.1*
3	*100*	*500*	*0.2*	*0.7*	*0.1*
4	*100*	*500*	*0.1*	*0.8*	*0.1*

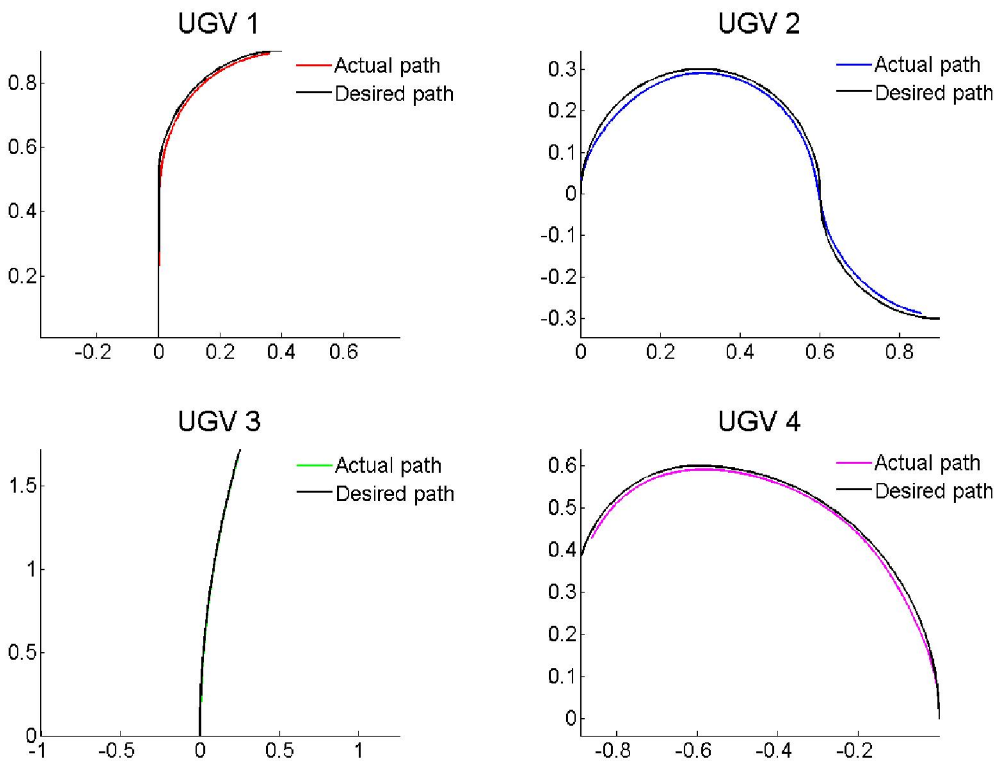

Figure 8.5 The path tracking results of case #2

Table 8-4 Nash Equilibrium Results of the CGA Process

Test Case #	**Performance Value**				**Security Value**
	UGV1	**UGV2**	**UGV3**	**UGV4**	
1	*0.0266*	*0.0668*	*0.0159*	*0.0428*	*1.0568*
2	*0.0265*	*0.0666*	*0.0162*	*0.0427*	*1.0591*
3	*0.0265*	*0.0669*	*0.0155*	*0.0427*	*1.0588*
4	*0.0266*	*0.0668*	*0.0158*	*0.0426*	*1.0596*

8.4.4 Simulation Results: Different Time Delays

To compare and evaluate the optimal bandwidth allocation calculated by the CGA ($BW_1=20.68\%$, $BW_2=55.46\%$, $BW_3=3.76\%$, $BW_4=14.1\%$, $BW_{sec}=6\%$) in different network environments, we run several comparison simulations with the average bandwidth setting ($BW_{1\text{-}4}=20\%$, $BW_{sec}=20\%$) under different time delay settings.

Table 8-5 shows the improvement in the performance and security values of the CGA's optimal bandwidth allocation compared to the average bandwidth allocation at different time delays. This value was computed as:

$$\textbf{\% Improvements} = \frac{P_{average} - P_{CGA}}{P_{average}} \times 100\% \,. \tag{79}$$

For each delay setting, we run the experiments 10 times to get the average results. As we can see, the optimal bandwidth allocation using the CGA has an impressive improvement in performance metrics while it still maintains the same security level compared to the other settings. From the simulation results, we conclude that the proposed CGA paradigm provides a satisfactory modeling and optimization scheme for the multi-agent performance and security trade-off problem on D-NCS.

Table 8-5 Percentage Improvement in Performance and Security When Using CGA Optimal Bandwidth Allocation

Delay	**Percentage Improvement**	
	Performance	**Security**
60ms	*85%*	*-0.36%*
200ms	*245%*	*-1.06%*
400ms	*500%*	*-0.47%*
600ms	*823%*	*-2.54%*

CHAPTER 9. CONCLUSION AND FUTURE WORK

9.1 Summary

D-NCS are vulnerable to cyber attacks since the compromised agents in the system can affect other agents and cause the distributed control algorithms (e.g., consensus algorithm) to fail. This thesis considers the secure distributed control problem in an unreliable D-NCS. The vulnerabilities of the conventional linear consensus algorithm have been investigated and several case studies including multi-robot systems and wireless sensor networks have been discussed. The malicious attacks have been modeled as exogenous inputs that compromise the behavior of agents in the D-NCS.

The secure distributed control algorithms that embed distributed security mechanism internally for leader-follower and leaderless consensus network have been proposed [104], which include four/five phases:

1) Detection Phase: detect the neighbors' misbehaviors relying only on each agent's local observations;
2) Mitigation Phase: adjust the consensus computation weights according to the neighbors' reputation values;
3) Identification Phase: identify and isolate the compromised agents;
4) Update Phase: update the reference state using the adjusted consensus computation weights to ensure the convergence of well-behaving agents.

5) Recovery Phase: remove the impact of misbehaving agents by applying recovery schemes: rollback recovery and excitation recovery.

The proposed algorithms allow all the well-behaving agents to reach the consensus state in the presence of misbehaving agents in D-NCS. The effectiveness of the proposed methods is validated and analyzed through case studies both in simulations and experiments.

The additional security mechanisms to protect D-NCS might sacrifice the performance of the system due to limited system resources. To address the performance and security trade-off problem of D-NCS, a trade-off model for showing the trade-off between system real-time performance and security with limited resources in D-NCS has been presented [37]. The quantitative performance and security metrics have been developed and combined in a tradeoff objective function with two weighting factors representing the preferences on performance and security. A paradigm of multi-agent performance-security trade-off optimization based on the CGA for D-NCS has been proposed. Simulation results in the networked DC motor system and iSpace system have demonstrated that the CGA can provide an effective way for performance-security trade-off analysis and optimization in D-NCS [99, 105, 106].

9.2 Directions for Future Research

Although this research has solved many secure control problems in D-NCS, some challenges in securing the distributed control computations still need to be addressed. Here, we highlight some future research directions requiring further investigations.

9.2.1 Secure Distributed Control in the Presence of Colluding Misbehaving Agents

The characteristics of colluding misbehaving agents can be summarized as:

- They often know the entire system's structure and parameters;
- They can apply multiple malicious control signals and corrupt multiple measurements simultaneously;
- They can perform unbounded computations.

Unlike the single misbehaving agent's scenarios, which only consider generic failures, colluding misbehaving agents can synchronize and launch the worst-case attacks. Their objective is usually to change/disrupt the physical state of the whole system. A synchronized attack on D-NCS network components and physical plants can create new and complex security vulnerabilities that cannot be addressed by securing each agent individually.

Tools from social networking [107, 108], game theory [109, 110], and electronic business [111, 112] to extract important information and features about the behaviors of colluding misbehaving agents in the D-NCS can be considered to develop corresponding mathematical models. These mathematical models may be used to design distributed control algorithms with provable operational security guarantees, thus enabling the design of more secure D-NCS through a control perspective. We also envision that control theory [39, 57] and spectral graph theory [113] may be used to develop appropriate intrusion detection and reputation evaluation schemes.

9.2.2 *Secure Asynchronous Distributed Control Algorithms*

In this research, we have considered synchronous distributed control algorithms with a fixed communications network topology. However, some D-NCS are highly dynamic and geographically distributed and do not have fixed topologies. They are also heterogeneous interconnected systems with some tightly coupled local nodes and some loosely coupled inter-area nodes with stringent resource constraints. Each node may have different dynamics. Thus, a control algorithm operating at an agent (e.g., a load or a generator) of the D-NCS (e.g., Smart Grid) should utilize information local to the agent and should not expect any static infrastructure. The algorithm should also be robust against the network's fast dynamic environments and utilize minimal computational and communications resources by performing a few logical operations per iteration, as well as require lightweight data structures, so that the scheme is scalable to large systems.

Recently, a special class of distributed asynchronous consensus algorithms, the gossip algorithm, which reach consensus through iterative pair-wise averaging and enable asynchronous operations, have received much interest [20, 114, 115]. The gossip algorithm is inspired by the form of gossip seen in social networks where the problem under study is highly dynamic, unstructured, and extremely large. Information is passing around only from neighbors to neighbors asynchronously as in the analogy of office workers spreading rumors. The gossip algorithm is a potential solution to many of the D-NCS' challenging control and management tasks. Thus, the fundamental properties, limitations and vulnerabilities of the gossip algorithm with the presence of misbehaving agents deserve investigations.

9.2.3 Applications in Smart Grid Secure Distributed Operations

In a smart grid environment, various renewable resources, storage, and loads are spatially distributed, and various events (e.g., fault, power-trading) can occur. To ensure effective control and management for such a spatial, temporal, and event-driven problem, a hybrid model including both time-based and event-based agents should be considered. The Smart Grid Simulator [65, 116, 117] developed in ADAC lab at NCSU can be used as a digital testbed to facilitate the investigation and demonstrate the effectiveness of the proposed secure distributed control algorithms for Smart Grids.

Appropriate objective functions and measures to properly reflect Smart Grid performance (e.g., line losses, stability, convergence rate of the system to reach a steady-state after perturbations, disturbances, and/or reconfigurations) and vulnerability to malicious cyber attacks based on information such as the network topologies, the parameter settings of the algorithm, and each agent's dynamics can be considered. Preliminary results in this area are discussed in [118, 119].

REFERENCES

[1] R. A. Gupta and M.-Y. Chow, "Networked Control System: Overview and Research Trends," *IEEE Transactions on Industrial Electronics,* vol. 57, pp. 2527-2535, 2010.

[2] "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.

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

[4] 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.

[5] S. Sridhar, A. Hahn, and M. Govindarasu, "Cyber-Physical System Security for the Electric Power Grid," *Proceedings of the IEEE,* vol. 100, pp. 210-224, 2012.

[6] A. Banerjee, K. K. Venkatasubramanian, T. Mukherjee, and S. K. S. Gupta, "Ensuring Safety, Security, and Sustainability of Mission-Critical Cyber-Physical Systems," *Proceedings of the IEEE,* vol. 100, pp. 283-299, 2012.

[7] J. Fink, A. Ribeiro, and V. Kumar, "Robust Control for Mobility and Wireless Communication in Cyber-Physical Systems With Application to Robot Teams," *Proceedings of the IEEE,* vol. 100, pp. 164-178, 2012.

[8] M. Mehyar, D. Spanos, J. Pongsajapan, S. H. Low, and R. M. Murray, "Asynchronous Distributed Averaging on Communication Networks," *Networking, IEEE/ACM Transactions on,* vol. 15, pp. 512-520, 2007.

[9] P. A. Bliman and G. Ferrari-Trecate, "Average consensus problems in networks of agents with delayed communications," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, 2005, pp. 7066-7071.

[10] M. Cao, D. A. Spielman, and A. S. Morse, "A Lower Bound on Convergence of a Distributed Network Consensus Algorithm," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, 2005, pp. 2356-2361.

[11] K. Yoonsoo and M. Mesbahi, "On maximizing the second smallest eigenvalue of a state-dependent graph Laplacian," *Automatic Control, IEEE Transactions on,* vol. 51, pp. 116-120, 2006.

[12] J. Meng and M. Egerstedt, "Connectedness preserving distributed coordination control over dynamic graphs," in *American Control Conference, 2005. Proceedings of the 2005*, 2005, pp. 93-98.

[13] M. Mesbahi, "On State-dependent dynamic graphs and their controllability properties," *Automatic Control, IEEE Transactions on,* vol. 50, pp. 387-392, 2005.

[14] M. M. Zavlanos and G. J. Pappas, "Controlling Connectivity of Dynamic Graphs," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, 2005, pp. 6388-6393.

[15] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: algorithms and theory," *Automatic Control, IEEE Transactions on,* vol. 51, pp. 401-420, 2006.

[16] R. A. Freeman, Y. Peng, and K. M. Lynch, "Distributed estimation and control of swarm formation statistics," in *American Control Conference, 2006*, 2006, p. 7 pp.

[17] S. Martinez, F. Bullo, J. Cortes, and E. Frazzoli, "On Synchronous Robotic Networks Part I: Models, Tasks, and Complexity," *Automatic Control, IEEE Transactions on,* vol. 52, pp. 2199-2213, 2007.

[18] S. Martinez, F. Bullo, J. Cortes, and E. Frazzoli, "On Synchronous Robotic Networks Part II: Time Complexity of Rendezvous and Deployment Algorithms," *Automatic Control, IEEE Transactions on,* vol. 52, pp. 2214-2226, 2007.

[19] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip algorithms: design, analysis and applications," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 2005, pp. 1653-1664 vol. 3.

[20] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *Information Theory, IEEE Transactions on,* vol. 52, pp. 2508-2530, 2006.

[21] H. G. Tanner, G. J. Pappas, and V. Kumar, "Leader-to-formation stability," *Robotics and Automation, IEEE Transactions on,* vol. 20, pp. 443-455, 2004.

[22] L. Zhiyun, B. Francis, and M. Maggiore, "Necessary and sufficient graphical conditions for formation control of unicycles," *Automatic Control, IEEE Transactions on,* vol. 50, pp. 121-127, 2005.

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

[24] R. O. Saber, W. B. Dunbar, and R. M. Murray, "Cooperative control of multi-vehicle systems using cost graphs and optimization," in *American Control Conference, 2003. Proceedings of the 2003*, 2003, pp. 2217-2222 vol.3.

[25] M. Alighanbari and J. P. How, "Decentralized Task Assignment for Unmanned Aerial Vehicles," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, 2005, pp. 5668-5673.

[26] E. Franco, R. Olfati-Saber, T. Parisini, and M. M. Polycarpou, "Distributed Fault Diagnosis using Sensor Networks and Consensus-based Filters," in *Decision and Control, 2006 45th IEEE Conference on*, 2006, pp. 386-391.

[27] J. Y. Chen, G. Pandurangan, and D. Xu, "Robust Computation of Aggregates in Wireless Sensor Networks: Distributed Randomized Algorithms and Analysis," *Parallel and Distributed Systems, IEEE Transactions on,* vol. 17, pp. 987-1000, 2006.

[28] D. P. Spanos, R. Olfati-Saber, and R. M. Murray, "Approximate distributed Kalman filtering in sensor networks with quantifiable performance," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, 2005, pp. 133-139.

[29] A. Boukerche, A. Zarrad, and R. B. Araujo, "A Cross-Layer Approach-Based Gnutella for Collaborative Virtual Environments over Mobile Ad Hoc Networks," *Parallel and Distributed Systems, IEEE Transactions on,* vol. 21, pp. 911-924, 2010.

[30] J. F. Chamberland and V. V. Veeravalli, "Decentralized detection in sensor networks," *Signal Processing, IEEE Transactions on,* vol. 51, pp. 407-416, 2003.

[31] S. Kar, S. Aldosari, and J. M. F. Moura, "Topology for Distributed Inference on Graphs," *Signal Processing, IEEE Transactions on,* vol. 56, pp. 2609-2613, 2008.

[32] F. Cattivelli and A. H. Sayed, "Diffusion LMS-based distributed detection over adaptive networks," in *Signals, Systems and Computers, 2009 Conference Record of the Forty-Third Asilomar Conference on*, 2009, pp. 171-175.

[33] D. Bajovic, Jakovetic, J. Xavier, B. Sinopoli, and J. M. F. Moura, "Distributed detection over time varying networks: Large deviations analysis," in *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, 2010, pp. 302-309.

[34] A. A. Cardenas, S. Amin, and S. Sastry, "Secure Control: Towards Survivable Cyber-Physical Systems," in *Distributed Computing Systems Workshops, 2008. ICDCS '08. 28th International Conference on*, 2008, pp. 495-500.

[35] C. Neuman, "Challenges in Security for Cyber-Physical Systems," in *DHS Workshop on Future Directions in Cyber-Physical Systems Security*, Newark, NJ, July 22-24, 2009.

[36] M. Amin, "Security Challenges for the Electricity Infrastructure," *IEEE Computer Security and Privacy Supplement,* vol. 35, pp. 8-10, 2002.

[37] W. Zeng and M.-Y. Chow, "A trade-off model for performance and security in secured Networked Control Systems," in *2011 IEEE International Symposium on Industrial Electronics (ISIE)*, 2011, pp. 1997-2002.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," in *Proceedings of 42nd IEEE Conference on Decision and Control*, 2003, pp. 4997-5002.

[43] J. C. Maxwell, "On Governors," *Proceedings of the Royal Society of London,* vol. 16, pp. 270-283, 1867.

[44] N. P. Mahalik and K. Kiseon, "A Prototype for Hardware-in-the-Loop Simulation of a Distributed Control Architecture," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 38, pp. 189-200, 2008.

[45] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control,* vol. 49, pp. 1520-1533, 2004.

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

[47] D. Dzung, M. Naedele, T. P. Von Hoff, and M. Crevatin, "Security for Industrial Communication Systems," *Proceedings of the IEEE,* vol. 93, pp. 1152-1177, 2005.

[48] D. T. H. J. Kim, K. B. Y. Mo, D. Dickinson, H. Lee, A. Perrig, and B. Sinopoli., "Cyber–Physical Security of a Smart Grid Infrastructure," *Proceedings of the IEEE, special issue on Cyber-Physical Systems,* pp. 1-15, December 2011.

[49] G. N. Ericsson, "Cyber Security and Power System Communication: Essential Parts of a Smart Grid Infrastructure," *IEEE Transactions on Power Delivery,* vol. 25, pp. 1501-1507, 2010.

[50] H. Khurana, M. Hadley, L. Ning, and D. A. Frincke, "Smart-Grid Security Issues," *IEEE Security & Privacy,* vol. 8, pp. 81-85, 2010.

[51] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," presented at the Proceedings of the 16th ACM conference on Computer and communications security, Chicago, Illinois, USA, 2009.

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

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

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

[55] N. H. Vaidya, L. Tseng, and G. Liang, "Iterative approximate byzantine consensus in arbitrary directed graphs," in *Proceedings of the 2012 ACM symposium on Principles of distributed computing*, Madeira, Portugal, 2012, pp. 365-374.

[56] F. Pasqualetti, A. Bicchi, and F. Bullo, "Distributed intrusion detection for secure consensus computations," in *46th IEEE Conference on Decision and Control*, 2007, pp. 5594-5599.

[57] 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.

[58] 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.

[59] D. Acemoglu, A. Ozdaglar, and A. ParandehGheibi, "Spread of (mis)information in social networks," *Games and Economic Behavior,* vol. 70, pp. 194-227, 2010.

[60] S. Buchegger and J.-y. L. Boudec, "Performance analysis of the confidant protocol: Cooperation of nodes - fairness in dynamic ad-hoc networks," in *Proceedings of the*

3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2002, pp. 226-236.

[61] 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.

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

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

[64] R. A. Gupta, A. A. Masoud, and M.-Y. Chow, "A Delay-Tolerant Potential-Field-Based Network Implementation of an Integrated Navigation System," *IEEE Transactions on Industrial Electronics,* vol. 57, pp. 769-783, 2010.

[65] W. Su, W. Zeng, and M.-Y. Chow, "A digital testbed for a PHEV/PEV enabled parking lot in a Smart Grid environment," in *2012 IEEE PES Innovative Smart Grid Technologies (ISGT)*, 2012, pp. 1-7.

[66] 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.

[67] T. Mukherjee and S. K. S. Gupta, "A Modeling Framework for Evaluating Effectiveness of Smart-Infrastructure Crises Management Systems," in *Technologies for Homeland Security, 2008 IEEE Conference on*, 2008, pp. 549-554.

[68] C.-H. Tsang and S. Kwong, "Multi-agent intrusion detection system in industrial network using ant colony clustering approach and unsupervised feature extraction," in *Industrial Technology, 2005. ICIT 2005. IEEE International Conference on*, 2005, pp. 51-56.

[69] A. Creery and E. J. Byres, "Industrial cybersecurity for power system and SCADA networks," in *Petroleum and Chemical Industry Conference, 2005. Industry Applications Society 52nd Annual*, 2005, pp. 303-309.

[70] X. Yuefei, S. Ronggong, L. Korba, W. Lihui, S. Weiming, and S. Lang, "Distributed device networks with security constraints," *Industrial Informatics, IEEE Transactions on,* vol. 1, pp. 217-225, 2005.

[71] R. A. Gupta, A. K. Agarwal, M.-Y. Chow, and W. Wang, "Performance Assessment of Data and Time-Sensitive Wireless Distributed Networked-Control-Systems in Presence of Information Security," in *Military Communications Conference, 2007. MILCOM 2007. IEEE*, 2007, pp. 1-7.

[72] R. A. Gupta, M.-Y. Chow, A. K. Agarwal, and W. Wang, "Information security with real-time operation: performance assessment for next generation wireless distributed networked-control-systems," in *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE*, 2007, pp. 420-425.

[73] H. Li, M.-Y. Chow, and Z. Sun, "EDA-Based Speed Control of a Networked DC Motor System With Time Delays and Packet Losses," *IEEE Transactions on Industrial Electronics,* vol. 56, pp. 1727-1735, 2009.

[74] H. Li, M.-Y. Chow, and Z. Sun, "Optimal Stabilizing Gain Selection for Networked Control Systems With Time Delays and Packet Losses," *Control Systems Technology, IEEE Transactions on,* vol. 17, pp. 1154-1162, 2009.

[75] C. Kaufman, R. Perlman, and M. Speciner, *Network Security: Private Communication in a Public World*. Englewood Cliffs, New Jersey: Prentice Hall, 1995.

[76] R. A. Gupta and M.-Y. Chow, "Performance assessment and compensation for secure networked control systems," in *Industrial Electronics, 2008. IECON 2008. 34th Annual Conference of IEEE*, 2008, pp. 2929-2934.

[77] Y. Zhai, P. Ning, P. Iyer, and D. S. Reeves, "Reasoning about complementary intrusion evidence," in *Computer Security Applications Conference, 2004. 20th Annual*, 2004, pp. 39-48.

[78] S. S. Yau, Y. Yin, and H. G. An, "An Adaptive Tradeoff Model for Service Performance and Security in Service-Based Systems," in *Web Services, 2009. ICWS 2009. IEEE International Conference on*, 2009, pp. 287-294.

[79] M. A. Haleem, C. N. Mathur, R. Chandramouli, and K. P. Subbalakshmi, "Opportunistic Encryption: A Trade-Off between Security and Throughput in Wireless Networks," *Dependable and Secure Computing, IEEE Transactions on,* vol. 4, pp. 313-324, 2007.

[80] M. A. Potter and K. A. D. Jong, "Evolving neural networks with collaborative species," *In Proceedings of the 1995 Summer Computer Simulation Conference,* pp. 340--345, 1995.

[81] M. A. Potter and K. A. D. Jong, "Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents," *Evol. Comput.,* vol. 8, pp. 1-29, 2000.

[82] R. P. Wiegand, "An analysis of cooperative coevolutionary algorithms," PhD Thesis, George Mason University, 2004.

[83] H. Chen, K. P. Wong, D. H. M. Nguyen, and C. Y. Chung, "Analyzing oligopolistic electricity market using coevolutionary computation," *Power Systems, IEEE Transactions on,* vol. 21, pp. 143-152, 2006.

[84] F. Careri, C. Genesi, P. Marannino, M. Montagna, S. Rossi, and I. Siviero, "Strategic bidding in a day-ahead market by coevolutionary genetic algorithms," in *Power and Energy Society General Meeting, 2010 IEEE*, 2010, pp. 1-8.

[85] Z. Skolicki, T. Arciszewski, M. H. Houck, and K. D. Jong, "Co-evolution of terrorist and security scenarios for water distribution systems," *Adv. Eng. Softw.,* vol. 39, pp. 801-811, 2008.

[86] C.-K. Goh and K. Chen Tan, "A Competitive-Cooperative Coevolutionary Paradigm for Dynamic Multiobjective Optimization," *Evolutionary Computation, IEEE Transactions on,* vol. 13, pp. 103-127, 2009.

[87] C. K. Goh, K. C. Tan, D. S. Liu, and S. C. Chiam, "A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design," *European Journal of Operational Research,* vol. 202, pp. 42-54, 2010.

[88] K. C. Tan, Y. J. Yang, and C. K. Goh, "A distributed Cooperative coevolutionary algorithm for multiobjective optimization," *Evolutionary Computation, IEEE Transactions on,* vol. 10, pp. 527-549, 2006.

[89] W. Su and M.-Y. Chow, "Performance evaluation of a PHEV parking station using Particle Swarm Optimization," in *Power and Energy Society General Meeting, 2011 IEEE*, 2011, pp. 1-6.

[90] W. Su and M.-Y. Chow, "Performance Evaluation of an EDA-Based Large-Scale Plug-In Hybrid Electric Vehicle Charging Algorithm," *IEEE Transactions on Smart Grid,* vol. 3, pp. 308-315, 2012.

[91] A. Ulusoy, O. Gurbuz, and A. Onat, "Wireless Model-Based Predictive Networked Control System Over Cooperative Wireless Network," *IEEE Transactions on Industrial Informatics,* vol. 7, pp. 41-51, 2011.

[92] E. Toscano and L. Lo Bello, "Multichannel Superframe Scheduling for IEEE 802.15.4 Industrial Wireless Sensor Networks," *Industrial Informatics, IEEE Transactions on,* vol. 8, pp. 337-350, 2012.

[93] L. Man, X. Li, L. T. Yang, Q. Xiao, Z. Nenggan, W. Zhaohui, and Q. Meikang, "Static Security Optimization for Real-Time Systems," *Industrial Informatics, IEEE Transactions on,* vol. 5, pp. 22-37, 2009.

[94] W. Granzer, F. Praus, and W. Kastner, "Security in Building Automation Systems," *Industrial Electronics, IEEE Transactions on,* vol. 57, pp. 3622-3630, 2010.

[95] P. Marti, L. Caixue, S. A. Brandt, M. Velasco, and J. M. Fuertes, "Draco: Efficient Resource Management for Resource-Constrained Control Tasks," *Computers, IEEE Transactions on,* vol. 58, pp. 90-105, 2009.

[96] M. Cheminod, A. Pironti, and R. Sisto, "Formal Vulnerability Analysis of a Security System for Remote Fieldbus Access," *Industrial Informatics, IEEE Transactions on,* vol. 7, pp. 30-40, 2011.

[97] F. Macia-Perez, F. Mora-Gimeno, D. Marcos-Jorquera, J. A. Gil-Martinez-Abarca, H. Ramos-Morillo, and I. Lorenzo-Fonseca, "Network Intrusion Detection System Embedded on a Smart Sensor," *Industrial Electronics, IEEE Transactions on,* vol. 58, pp. 722-732, 2011.

[98] Y. Xu, R. Song, L. Korba, L. Wang, W. Shen, and S. Lang, "Distributed device networks with security constraints," *Industrial Informatics, IEEE Transactions on,* vol. 1, pp. 217-225, 2005.

[99] W. Zeng and M.-Y. Chow, "Optimal Trade-off between Performance and Security in Networked Control Systems based on Coevolutionary Algorithms," *IEEE Transactions on Industrial Electronics,* vol. 59, pp. 3016 - 3025, 2012.

[100] K. Yoshizawa, H. Hashimoto, M. Wada, and S. Mori, "Path tracking control of mobile robots using a quadratic curve," in *Intelligent Vehicles Symposium, 1996., Proceedings of the 1996 IEEE*, 1996, pp. 58-63.

[101] B. R. Klingenberg, U. Ojha, and M.-Y. Chow, "Predictive control of multiple UGVs in a NCS with adaptive bandwidth allocation," in *Industrial Electronics, 2009. IECON '09. 35th Annual Conference of IEEE*, 2009, pp. 3027-3032.

[102] U. Ojha and M.-Y. Chow, "Realization and validation of Delay Tolerant Behavior Control based Adaptive Bandwidth Allocation for networked control system," in *2010 IEEE International Symposium on Industrial Electronics (ISIE)*, 2010, pp. 2853-2858.

[103] J. W. Friedman, "A noncooperative equilibrium for supergames," *Review of Economic Studies,* vol. 38, pp. 1-12, 1971.

[104] 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.

[105] 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.

[106] W. Zeng and M.-Y. Chow, "CGA based performance-security trade-off optimization in a networked DC motor system," in *2012 IEEE International Symposium on Industrial Electronics (ISIE)*, 2012, pp. 1834-1839.

[107] M. E. J. Newman, "The Structure and Function of Complex Networks," *SIAM Review,* vol. 45, p. 167, 2003.

[108] M. E. J. Newman, *Networks : an introduction*. Oxford ; New York: Oxford University Press, 2010.

[109] Y. Dai, X. Chao, S.-C. Fang, and H. Nuttle, "Game Theoretic Analysis of a Distribution System with Customer Market Search," *Annals of Operations Research,* vol. 135, pp. 223-228, 2005.

[110] Z. Sheng and W. Fan, "A Collusion-Resistant Routing Scheme for Noncooperative Wireless Ad Hoc Networks," *Networking, IEEE/ACM Transactions on,* vol. 18, pp. 582-595, 2010.

[111] P. Resnick, R. Zeckhauser, J. Swanson, and K. Lockwood, "The value of reputation on eBay: A controlled experiment," *Experimental Economics,* vol. 9, pp. 79-101, 2006.

[112] A. G. P. Rahbar and O. Yang, "PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing," *IEEE Transactions on Parallel and Distributed Systems,* vol. 18, pp. 460-473, 2007.

[113] F. R. K. Chung, *Spectral graph theory*. Providence, R.I.: Published for the Conference Board of the mathematical sciences by the American Mathematical Society, 1997.

[114] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, "Randomized rumor spreading," in *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, 2000, pp. 565-574.

[115] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, 2003, pp. 482-491.

[116] Y. Xichun and M.-Y. Chow, "A Spatial, Temporal and Event Simulator of a Smart Grid," in *Poster Session*, ed. PES General Meeting, Detroit, MI, USA, 2011.

[117] W. Su, E. H., W. Zeng, and M.-Y. Chow, "A Survey on the Electrification of Transportation in a Smart Grid Environment," *IEEE Transactions on Industrial Informatics,* vol. 8, pp. 1-10, 2012.

[118] O. Kosut, J. Liyan, R. J. Thomas, and T. Lang, "Malicious Data Attacks on the Smart Grid," *IEEE Transactions on Smart Grid,* vol. 2, pp. 645-658, 2011.

[119] F. Pasqualetti, R. Carli, and F. Bullo, "Distributed estimation via iterative projections with application to power network monitoring," *Automatica,* vol. 48, pp. 747-758, 2012.