SYSTEM SECURITY, THREAT DETECTION AND PREVENTION MEASURES OF AUTONOMOUS SYSTEMS

by

CHAITANYA RANI VEERANNA GOWDA

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2015

战争 **的 最高** 艺术 **是 不**战 **而 屈人之兵**.

[Zhanzheng de Zuigao Yishu Shi Buzhan Er Qurenzhibing]

"The supreme art of war is to subdue the enemy without fighting"

-Sun Tzu, The Art of War

Acknowledgements

I am grateful to Dr. Frank Lewis for giving me an opportunity to work with him and mentoring my Master's Thesis. I would like to thank Dr. Aditya Das for his constant guidance and being patient in helping us write papers. I thank Dr. William Dillon for serving on my committee.

My sincere thanks to Honghai for spending valuable time in research.

I thank Meghana, Pallak, Rushil and Sourav for helping me get back on my feet and their succor despite the differences in time zones. Raghu for his motivation, Abhinav for his honest and meticulous inputs, Rohit Rawat for his invaluable help in debugging codes. Auddy, Nayana, Dilip for their support and timely feedback.

Most importantly, I thank Paa, Maa and Guru for believing in my abilities and aiding in pursuing my dreams. Ajji, Thata, Satish Mama and Kalpana Aunty for being the greatest teachers and making childhood a memorable experience.

April 17, 2015

Abstract

SYSTEM SECURITY, THREAT DETECTION AND PREVENTION MEASURES

Chaitanya Rani Veeranna Gowda, M.S

The University of Texas at Arlington, 2015

Supervising Professor: Frank L. Lewis

The Federal Aviation Administration (FAA) estimates that by the year 2020, the United States will have over 30,000 drones. Today, the fields of utilization of drones, also known as Unmanned Aerial Vehicles (UAV) are unlimited. UAVs can be used in hazardous military missions, since they exclude the risk factors involved in manned vehicles. The numerous sensors on board gather data related to the desired flight plan. This exposes UAVs to various vulnerabilities since they carry enormous information and raises safety issues and privacy concerns. Absence of manual control by a human operator over the UAV results in fraudulent information being fed and navigated to a different location by the attacker. On gaining control of the system, sensitive data can be accessed and misused. The objective of this research is to study, develop and implement various security, threat assessment and response systems on autonomous systems.

Table of Contents

List of Illustrations

Chapter 1

Introduction

UAVs have gained popularity since they can be controlled remotely or programmed to have an autonomous flight mission. With the advancement of technology, the duties of a pilot can be simulated on a low level by using various built-in guidance, navigation and control systems on board. They can be used in areas where human supervision is perilous and inaccessible.

Monitoring sites of gas leakage, forest fires, pipelines in factories, underground mines and caves, weather monitoring of storms and hurricanes, to measure pollution levels, military training, volcanic gases, disaster relief [1][2]. Under severe conditions UAVs can be used in missions to search for survivors and rescue them[3]. Additional sensors such as night vision cameras, thermal sensors, GPS and lidars can be mounted on board to collect data from the investigation site and the operator can intervene from a remote site.

Applications of UAVs in the field of photography/cinematography and commercial purposes are plenty. Amazon [4] announced the utilization of UAV technology to deliver packages of customers under thirty minutes. The rise in cyber criminal activity must be contemplated before the lives of civilians is in danger. Use of microlight aerial vehicles could cause potential security breach in privacy of civilians. They can carry small lethal payloads.

Most UAVs use communication over a wireless network to transmit information to and from the operator on the ground. Wi-Fi, GPS, Bluetooth, Infrared, and ZigBee are some forms of wireless communication. These are easily susceptible to attacks. A company used a drone to film an Australian triathlon, the operator lost complete authority over the vehicle and crashed into one of the athletes causing injuries [5]. It was reported that an attacker had manipulated the communication link of the drone's control by a "channel hop" attack. Simple Wi-Fi and GPS jammers can be used to perform such attacks. One such UAV is the Parrot AR.Drone, which operates on an unsecured Wi-Fi network and access to the drone can be obtained easily. This

possibility is explored in Section 5 of this thesis. GPS spoofing [6] is another threat that fools the receiver to track fraudulent signals by providing falsified information about the location. Bilateral verification processes done by receiver and transmitter could possibly avoid such interferences.

UAVs will populate the sky in the next few years for various purposes, and effective laws have to be set to counter exploitation of this technology by foreign countries and cyber attacks within the country. Security of civilians should be of prime importance. This thesis highlights the importance of security measures in UAVs. It provides a general overview of current hacking methods, prevention, implementation of trust strategies and control measures to overcome the problems related to cyber attacks involving UAVs.

Sensepoint, London-based security firm set up a spy software named 'Snoopy'. It is a software running on a UAV looking for smartphones connected to open Wi-Fi networks in airports, cafes and train stations. It receives user's information off of the smartphone such as usernames, passwords and credit card details [7]. An example of a software tool hacking a drone is explored in this thesis. Section 2 introduces various hacking strategies to exploit a system's ability. Section 3 talks about the numerous strategies to protect a system against an attack due to hacking. Section 4 outlines the use of Control System and Graph Theory strategies to implement Trust-based control in Unmanned Systems. Section 5 enunciates an experiment conducted using the Parrot AR.Drone and a Wi-Fi cracking software tool to gain access over the system completely.

Chapter 2

Hacking methods

Hacking refers to gaining illegitimate entry into another system or network by illegal methods. Wireless attacks are the most common form of hacking. In this section, we deal with various hacking strategies to compromise a system's ability to be controlled by its rightful owner. The techniques used are :

a) Password Theft

b) Wireshark

c) Man-In-the-Middle Attacks

d) Trojan Horse Virus

e) Distributed Denial of Service (DDoS) Attacks

2.1 Password Theft

Passwords are usually a sequence 8 to 16 character combinations formed using the keyboard. Unique passwords have to be created by blending upper and lower case alphabets, numbers and special characters to ensure safety. This leads to complex passwords which are not easy to recall. Unfortunately, even complex passwords can be cracked using software tools such as dictionary attacks, brute force attacks and statistical methods such as Aircrack-ng.

Aircrack-ng is a WEP (Wired Equivalent Privacy) and WPA-PSK (Wi-Fi Protected Access Pre-Shared Key) key cracking program [8]. It is a collection of tools to crack passwords. Airmon-ng puts our wireless card from managed mode to monitor mode. Necessary drivers for the chipset have to be installed for this to work. It enables the network card to view all the traffic. Airodump-ng allows us to capture packets of a particular client out of the list of clients available. Frames are injected using the Aireplay-ng tool.

It has a wide variety of applications such as de-authentication, fragmentation and cafe-latte attack [9]. Use of Aircrack-ng is further elaborated in Section 5 to gain access to a UAV platform such as the Parrot AR.Drone.

2.2 Wireshark

Wireshark is a robust tool to analyze and capture packets for wireless networks compatible on Linux, Windows and Mac. The origin and target are easily identified and provides valuable and sensitive information of the transmitted packets between them. This allows effortless access to a client system and gain control over it. Once wireshark is put to promiscuous mode, a list of available interfaces are displayed [10]. A victim's username and password can be obtained using 'find' option on the GUI. Wireshark is useful when websites use HTTP connections. Traffic of Voice over Internet (VoIP) calls and raw USB can also be captured. It fails when HTTPS protocol is used.

2.3 Man-In-The-Middle (MITM) Attacks

Man-In-The-Middle is a type of attack where the attacker gains control of sensitive data by furtively modifying the communication link between two parties. The end users are usually unaware of the manipulation performed by the attacker. Figure 1 further illustrates the definition of a MITM attack.

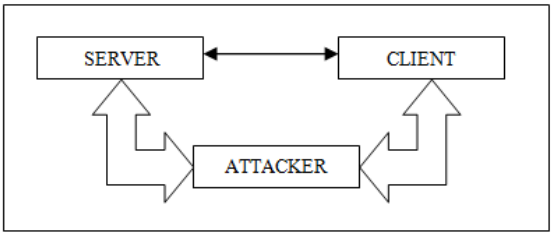

Figure 2-1 The Man-in-the-middle is the attacker forming a fake connection between the server and the client

A simple example of a MITM attack is scam e-mail disguised as a genuine e-mail that misguides the user to fake site. The user is then tricked to login while the attacker eavesdrops and

collects credential information such as passwords, user name and credit card numbers [11]. Other forms of MITM attacks are URL manipulation, rogue DNS (Domain Name Server) and ARP (Address Resolution Protocol) poisoning, duplication of MAC (Media Access Control) [12]. To prevent such attacks, only trusted websites and domains should be accessed.

2.4 Trojan Horse Virus

Trojan Horse Virus is a malicious program or software that causes detrimental effects such as monitor traffic over a network, destroy files and damage hard drives in a system. Such viruses usually take leverage of a security glitch and multiply quick. It gives the attacker an access to the system remotely. This is known as a backdoor Trojan [13]. Unwanted e-mail attachments and downloads could fool the user to open the link or install the software. A destructive Trojan can continuously delete all files and ultimately demolish the Operating System. These viruses cannot be easily identified by anti-virus programs. Mac OS and Linux are usually not affected by the Trojan virus.

The best way to prevent Trojan attacks would be to minimize risk at the root cause. Unsolicited e-mails should not be entertained and a good quality anti-malware installed in the system before being affected.

2.5 Distributed Denial of Service (DDoS)

A DDoS attack is a large-scale intrusion method performed by a host source which causes detrimental effects to legitimate users by withholding services [14]. This either causes the system to shut off completely or drain the system resources such as computing power and bandwidth of the victim rapidly. Exposed or defenseless computers are first targeted by virulent attackers. They generally have expired or no anti-virus software installed.

Once the attacker gains access to the system, new tools are installed to enable control of the host. Infected systems continue to look for other vulnerable systems and attack them.

This results in a master-slave process where the affected victim is being controlled by the attacker. Figure 2 illustrates this process of a DDoS attack. The attacker is now in possession of controls of a large number of systems. Resources of the system are exhausted rapidly and flooding of packets occurs on the victim end. There are various attack techniques : random scanning, local subnet scanning, hit-list scanning, permutation scanning, topological scanning [14].

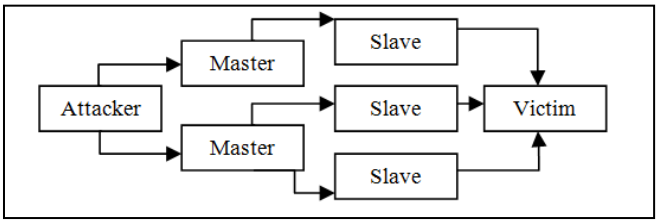

Figure 2-2 A Conventional DDoS attack

The attacker then removes all traces that could lead back to the source of the attack by using a spoofed IP (Internet Protocol) address, thereby preventing the victim to permeate any illegal traffic targeted towards them.

Steps in initiating a DDoS attack [15] :

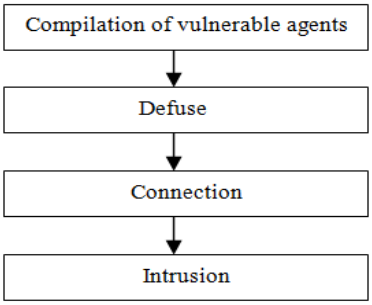

Figure 2-3 Steps in initiating a DDoS Attack

2.5.1.Compilation of vulnerable agents

The network is first scanned for vulnerable agents for the attacker to compile a list of agents to attack. Earlier, manual attempts were made to attack a system. In the recent years, due to

advancement of technology, it is possible to attack systems by setting up automated software to scan the network and take over vulnerable agents.

2.5.2. Defuse

Flaws in security and agent vulnerability are misused by the attacker. Software codes are used to automatically attack and disband the rightful owner from controlling the system. Suitable actions are taken by the attacker to safeguard the code planted in the system from being removed.

2.5.3. Connection

Protocols such as TCP or UDP is used to connect with numerous agents and plan attacks accordingly via scheduling. Attacks can be performed on either single of multiple agents.

2.5.4. Intrusion

The features of the victim such as port numbers, TTL (Time To Live), are conformed. The attacker then launches the attack and alters the properties. This is in favor of the attacker since alteration of the packets would lead to complications in identification of the source of attack.

Chapter 3

An Application Study: Hacking The Parrot AR.Drone

Here, we analyze the case of Unmanned Aerial Vehicles (UAVs) such as the Parrot AR.Drone. The Parrot AR.Drone is a quadcopter equipped with four rotors and it spawns its own Wi-Fi network allows users to connect to it and control. It functions on a 802.11 Wi-Fi protocol which is coupled with an authentic client. By enabling the packet capturing feature, we can gain control over the AR.Drone. Such devices are prone to being attacked. The Parrot AR.Drone can be controlled by users by downloading an application on their phones or tablets. It is available in iOS and Android platforms. Unofficial applications are available on Windows and Symbian phones.

The AR.Drone has frontal and base cameras to enable the user to access video streams of the drone's view on their devices. It has gained popularity due to mass production of the product leading to low cost availability. Hence it has been used for test loopholes and issues in security of UAVs. Here, we hack and control the Parrot AR.Drone using tools such as Aircrack-ng and Robot Operating System (ROS).

The AR.Drone connects over a Wi-Fi network to a user's device. The standards on 802.11 Wi-Fi makes it vulnerable to attacks caused by de-authentication. Today, the security in Wi-Fi includes WEP (Wired Equivalent Privacy) , Wi-Fi Protected Access (WPA) and Wi-Fi Protected Access II (WPA2). They can be broken into effortlessly. Here we use Aircrack-ng to de-authenticate a valid client and gain control over the system. Private or unseen SSID (Service Set Identifier) can also be retrieved.

Wired Equivalent Privacy (WEP) or Wi-Fi Protected Access (WPA) for scrutinizing wireless networks. All Wi-Fi signals have information about SSID of the network and MAC address of the device [40]. An Access Point (AP) enables a wireless device to connect to a

wireless network. Typically, the access point of the device connects to a router and has a range of 35-100 meters.

There three frame types in Wireless Networks, namely : data packets, management packets and control packets. An example of data packet is IP (Internet Protocol) packets. Management packets usually deal with the devices associating with an access point. Control packets aid in mediation.

There are three addresses linked to Wi-Fi networks, namely : BSSID (Basic Service Set ID) that indicates the Access Point used, source address which identifies where the packet is coming from, and destination address denotes the terminal point of the packet.

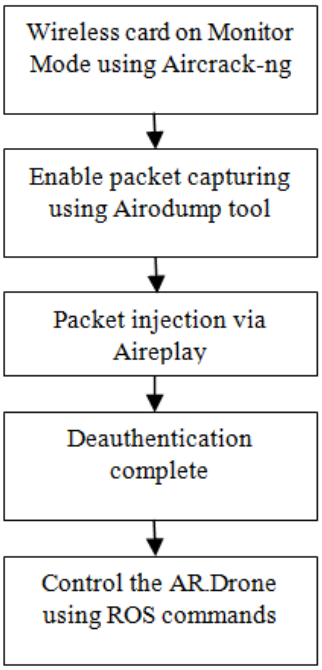

Figure 3-1 Steps in initiating an attack on the Parrot AR.Drone using Aircrack-ng and ROS

A Unix shell script was created using Aircrack-ng, a cracking program which acts as network sniffer and cracks the privacy component. Figure 3-1 illustrates the steps in initiating the attack. Aircrack-ng has a number of tools that allow hacking a device [41]. The airmon-ng

command enables monitor on the laptop's wireless card. For this purpose, the Alfa Wi-Fi adapter is used. It allows the wireless network to enable monitor mode as well as inject packets. Using the Alfa Wi-Fi adapter, MAC addresses can be identified. [42] The wireless card is first changed from managed mode to monitor mode inspect the traffic on wireless networks. Airodump-ng is a tool by Aircrack to discern all the wireless networks in the scope of the wireless card. The access points, channel numbers, BSSID of all the available stations are shown. The access point of the particular network that is chosen, can be blocked by Aireplay-ng. De-authentication packets are sent to allow the disconnection from the valid client. The '-a' term deals with the MAC address of the Parrot AR.Drone. '-c' deals with the Access point of the device used to control the drone. In this case, an android phone was used to control the drone. The 'ignore-negative-one' command was added to eliminate the issue caused by the wireless card. The card constantly looks for channels with the number -1, this would hinder the de-authentication attack. To fix this, the 'ignore-negative-one' was added. The script was run on a Linux machine, since the options for Aircrack-ng and the Alfa Wi-Fi adapter on a Windows platform was relatively lesser.

The MAC address of the Parrot AR.Drone is identified by enabling monitor mode on the Alfa Wi-Fi adapter. ESSID (Extended Service Set Identification) of the AR.Drone is noted by the malicious client and it forcefully connects to the laptop after the de-authentication has been successful from the valid client.

The control of the drone is done using ROS (Robot Operating System). It is an open-source platform that acts as a miniature operating system for a device. It has various tools to build and run codes. It is typically used on a Unix platform. The version used here is Ubuntu 12.04, since the development for this version is stable. ROS is also supported on other Linux platforms. It is currently not supported on a Windows Operating System [43] . The ROS driver used for Parrot AR.Drone is called 'ardrone_autonomy'. This was developed by the Autonomy lab at Simon Fraser University.

The drone can be controlled by publishing commands over the Wi-Fi network. 'Ardrone_autonomy' has documentation for joystick control, camera, motor control etc., of the AR.Drone. Currently, the command for land is published and drone acts accordingly. Using a joystick controller, the buttons can be tailored to perform various tasks for maneuvering the drone. Commands for take-off, land, left, right, up, down and flip can be given by the joystick controller and the drone is made to perform the tasks accordingly. The flight path can be adjusted according to the malicious client. The video feed of the drone's view can also be obtained using the tools in Ardrone_autonomy.

The MAC (Media Access Control) address is string of characters which is a combination of alphabets and numbers. An example of MAC address is 29:13:F4:5K:45:M1. This is unique to every network interface and is used to transmit and receive information. It i mostly used for identification purposes. The Alfa Wi-Fi adapter and Airodump-ng tools aid in monitoring the MAC addresses of the available AR.Drones in the range of the wireless card. A demonstration of attack was implemented successfully. The vulnerabilities of Parrot AR.Drone proves that many other UAVs are also capable of being prone to such attacks and arises the need to secure them.

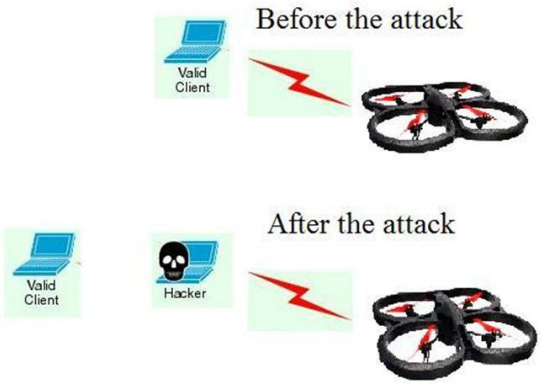

Figure 3-2 System architecture illustration before and after the attack

Chapter 4

Defense against Hacking

There are numerous methods to secure a system, by hampering the threat at the root or by identifying them and taking suitable actions when encountered. A system can never be completely secure, it will always be prone to some form of attack either internal or external. Sometimes the operating system could have flaws which can lead to vulnerability. This section provides an overview of a few existing solutions to aid security of a system. They are :

a) Encryption

b) Defense against DDoS

c) Intrusion Detection Systems (IDS)

4.1. Encryption

It is a simple method of securing sensitive data on systems. Encryption deals with encoding of data or message so that only legitimate users can gain access to it. It acts as a barrier to illegal activity by denying entry to secure information such as credit or debit card data, personal details and social security numbers. This enables governments and military organizations to promote secure transmission of sensitive information.

Civilians can use encryption keys to guard confidential data regarding personal records and protect themselves against fraud activity. Encryption can be utilized to insulate data being transferred over a network such as Wi-Fi, Bluetooth, intercom systems, mobile phones and ATMs (Automatic Teller Machines) in banks [16].

Although brute force attacks beat the purpose of encryption, the key length increases the strength of encryption. Another useful approach to inhibit hacking is to hide the access points by disabling SSID (Service Set Identifier) or by granting access to systems with familiar MAC addresses. Sometimes spoofed MAC addresses also enable hackers to join the network. In the

case of a UAV, a password can be used to help authenticate messages being communicated between UAV and the operator.

4.2. Defense against Distributed Denial of Service (DDoS) Attacks

There are several mechanisms to prevent DDoS, such as reactive and preventive mechanisms. Reactive method uses a mechanism to identify the attack at source and tries to inhibit the damage caused by it. They are also called 'Early Warning Systems' [14]. Preventive methods exclude the feasibility of the attack by taking remedial measures. Routine scanning by virus scanners, firewalls, patches, anti-malwares and maintaining appropriate protocols using sensors. These filtering mechanisms will help keep a check on abnormal activity. They maintain a log of normal behavior of the system and constantly compare to check for unusual or peculiar behaviors. A threshold is set to decide the factor of abnormality.

Signature based detection is also used to prevent DDoS attacks. A database of well known attacks are matched for signatures or patterns. The disadvantage of this method is that it cannot identify any new attacks on the system. Attempts to safeguard a system before being attacked is extremely important as it decreases the chances of a potential DDoS attack.

A hybrid system uses the combination of preventive and reactive methods as a countermeasure for DDoS. Reactive mechanisms can counter the behavior of an attack once it is detected [14]. DDoS exhausts the resources of the victim within a few seconds. To implement a faster technique to protect against DDoS would require a larger processing source. Although, this will affect the accuracy of the identification process.

4.2.1. Generic architecture of DDoS attack Defense mechanisms

To elaborate on the above defense mechanisms, three classes of DDoS defense mechanisms can be generalized as given in [17] : source-end, victim-end and intermediate defense.

- Mechanism of Source-end defense

Rate limiters are used to compare the traffic against a set of predefined profiles. Preventing an attack at the source-end is the best method. It avoids flooding in the network. Identifying attacks at the source end is usually a difficult task since malicious traffic may pretend to behave similar to normal traffic and this poses a challenging problem to the classifier of the system.

- Mechanism of Victim-end defense

Intrusion detection schemes are employed to detect malicious traffic. Signatures of abnormal activity are noted in a list and the behavior is constantly compared with other incoming or outgoing traffic. A threshold is set and notifies when misleading alarms are identified. The drawback of this mechanism is that it is too late for the attack to be prevented. This denies legitimate users the resources of the system.

- Mechanism of Intermediate defense

Intermediate defense combines the efforts of both source-end and victim-end defenses. Identifying the attack at source and using routers to deploy this information to prevent attack at the victim-end. Failure of routers or the detection mechanism at the source-end could have detrimental effects on the victim which completely relies on the source for initial defense.

A recommended method for defense against DDoS is DefCOM (Defensive Cooperative Overlay Mesh). DefCOM is a conglomerate of nodes that are formulated in a network where each system can convey information directly to other systems without having to use a central system for communication. Such a network is known as peer-to-peer.

Figure 4 from [18] gives a better understanding of the working of a DefCOM framework.

These frameworks typically consist of three nodes, namely : Alert generator, Rate limiter and Classifier.

- Alert Generator

An alert generator monitors the network to find probable virulent traffic and notifies all the other nodes. It must be able to detect when an attack can be expected. Firewalls and intrusion detection systems are good examples of alert generators. They are usually installed at the victim-end to detect any abnormal activity with high precision.

- Rate Limiter

Rate limiters are installed at the edge of the network to allow only selective traffic by comparing it against the list of legitimate traffic on the network. It prioritizes the flow of traffic to valid users and restricts the flow to suspicious nodes. Routers are used to implement rate limiters. They are beneficial where classifiers cannot be installed.

- Classifier

Classifiers are present at the core of the network to identify authentic traffic from virulent ones and uses rate limiters to limit the rate of traffic to suspicious users. The information from alert generators are utilized secure the system from any malicious activity. Potential destruction of the system can be reduced.

DefCOM does the work of identifying the attack on the source-end by using classifiers, rate limiters are used at the victim end to strain malicious traffic and the system is alarmed using alert generators.

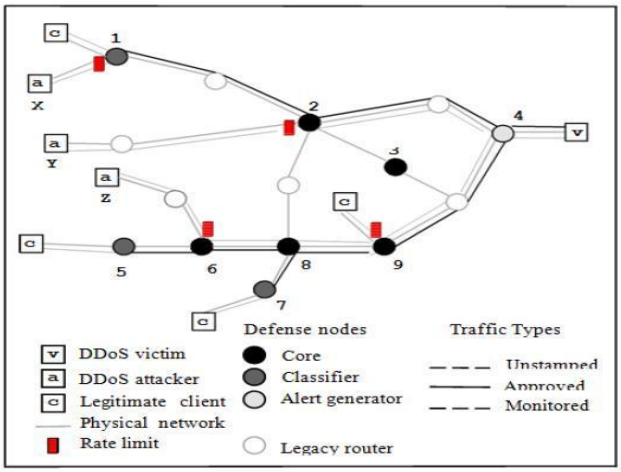

Figure 4-1 Illustration of Defensive Cooperative Overlay Mesh (DefCOM) framework

4.2.2. DDoS Defense using Soft Computing Methods

DDoS attacks can also be detected using Soft Computing methods as established in [18]. By using Artificial neural networks, fuzzy systems, genetic algorithms etc. analysis of DDoS attacks can be performed with intelligent techniques and classified automatically. Soft Computing is a an efficient way to process the uncertainty in the system.

Neural Networks can be used to evaluate and categorize the incoming or outgoing traffic as malicious or normal traffic. Radial Basis Functions (RBF) can be used to perform feature classification. Classifiers at the edge of victim-end are employed to trigger RBF networks. It uses a filtering technique to send malicious traffic to alert generators and normal traffic is sent to its destination.

4.3. Intrusion Detection Systems (IDS)

Intrusion is the act of wrongful entry or compromise of a system's resources without consent of the rightful owner. Systems can be tricked into allowing access to an intruder. Often, due to glitches in the working of the system, it may cause intrusions. Intrusion detection is the course of supervising and identifying indications of abnormal activity.

Intrusion Detection Systems (IDS) is usually done by a software that inspect the working of the system to find anomalies. They are defensive tools but do not provide preventive

actions to safeguard a system. There are two kinds of intrusion detection systems : Misuse IDS and Anomaly IDS.

Misuse IDS is given by pre-defined attack signatures that take advantage of security loopholes. Such signatures are well-known prior to the attack and are used to test against incoming patterns for its virulent nature. Anomaly IDS uses system's regular performance to understand its behavior and forms statistics accordingly. If there is a deviation from the normal usage performance then such a behavior is noted by the mechanism in an IDS.

Intrusion Detection Systems identify the incoming traffic and determines whether it has to be protected or not. These systems utilize three types of information : long-term, configuration and audit information.

Long-term information deals with building a database of techniques used in identifying attacks [19]. Configuration information relies on the present condition of the system. Audit information portrays the situation and circumstances of the system. Unwanted data from the audit and an artificial map of the state of the system is provided. Based on this, a conclusion is reached on whether the identified symptoms are an actual intrusion or not. Preventive measures are taken to bring the system back to a safe state.

By the use of firewalls, encryption and authentication the first barrier to secure a system is formed. As discussed earlier, password with shorter lengths or weak passwords are easily attacked. Firewalls are prone to errors while setting them up and may not guarantee safety of the systems at all times. Hence, there is a need for Intrusion Detection Systems to safeguard a system from attackers. They provide additional security along with the first barrier of protection by encryption, authentication and firewalls. Figure 5 illustrates the use of IDS in a network. Intrusion Detection Systems also provide safety measures to prevent attacks on a timely basis.

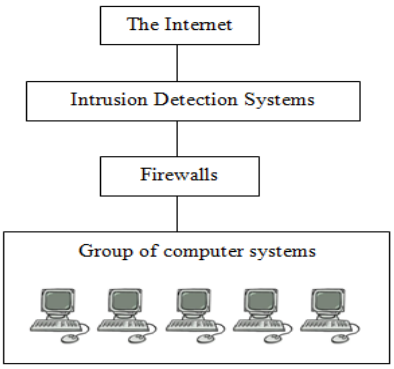

Figure 4-2 Illustration Detection System used as a Defense mechanism

4.3.1 Neural Networks in Intrusion Detection Systems

The application of Neural Networks in machine learning is a result of inspiration from biological neural systems. Individual neurons convey information to form a process [19]. Computation using neural networks is a novel method to determine solutions to problems in machines.

The structure of highly connected neurons act as centers of processing data. They contain a set of inputs that pass through a set of processing nodes to obtain the outputs. The information is studied and memorized by neurons that are interconnected, weighted synapses and threshold logic units in the network. Neurons are the processing elements which are computation components. These components are called summing elements. Activation functions are preceded by a summing component [20]. The input to neuron in one layer is derived from the output of a neuron in another layer.

Using classification functions, neural networks are able to perform intrusion detection. The neural net identifies the pattern or signature of the input and places it into classes accordingly at the output. Using neural networks, IDS is basically of two types : A) Multi-layer feed forward neural networks which consists of a hidden layer in between the input and output layers. B) Kohonen's Self-organizing maps which forms a mapping from input to the clusters.

The detection process is expedited with the use of a neural network since they have the ability to learn the features of past intrusions.

4.3.2. Fuzzy Rule Based Systems in IDS

Fuzzy logic is method that uses approximation techniques instead of a fixed values. In Binary systems, either 0 or 1 is used but with the help of fuzzy logic, we can derive ranges that lie between 0 and 1. This provides partial or intermediate values and hence fuzzy databases can be formed [21]. Using fuzzy systems, it is easy to map output space from a given input space. These systems are flexible and accommodate approximate values. Any valid set of data given based on input and output values, can be translated to form a fuzzy system. They are agents of simplification.

Fuzzy logic allows room for errors or uncertainty and provides low-cost solution in a system. It provides significant advantage over other soft computing methods in intrusion detection. It detects abnormal activity in the system and formalizes strategies to resolve it accordingly. In [22] the system used to detect intrusions is as follows :

A) Training data classification, B) Fuzzy rule strategy creation, C) Decision module using a Fuzzy rule, D) Classifying the input accordingly. The results of this provided more accuracy in the identification of anomalies based on the rules used in Fuzzy logic.

By utilizing approximate data, if-then rules can be formed with the aid of expert knowledge. The drawback in this method the number of if-then rules grows rapidly as the data set becomes larger. In Neural networks, training is required but in Fuzzy based system rules are formed to reach conclusions pertaining to a data set.

Chapter 5

Trust in System Security

This section deals with novel ways to secure a system based on graph theory and control system strategies. Trust propagation in networks is discussed to provide an insight on consensus algorithms. The framework of agents placed in the graph strongly influences the behavior of the agents and flow of communication between the nodes in the graph. Data from neighboring nodes is of great significance. Team based trust propagation methods in sensor networks and UAVs has been developed. Based on the nature of the nodes and the trust values each of them possess, it can be classified. Here we discuss a few strategies that have been previously used to secure a network.

In [23], to eliminate malicious agents, the method of negative trust values has been employed. When a node behaves erratically, the trust value of that particular node is updated adversely. Using game theory securing the system against malignant nodes was studied in [24]. On UAV platforms the concept of cooperative control is discussed in [25] by the implementation of a planner and learner that follow a safe policy initially and further move onto avert risky behavior by employing stochastic risk models. Such systems help raise awareness in networks.

Active Directory, a service on Windows networks developed by Microsoft. It checks the authenticity of the user's passwords and names to be capable of enjoying administrator rights on a system. It uses a network of trust relationships to allow sharing of sensitive data between domains [26]. The default trust types are parent and child, tree-root.

Using Game Theory, trust dynamics can be derived for cooperative control by [27] and [28]. Various trust games are defined : Additive, constant-sum, super-additive and convex to form trust models. This was applied to a military convoy where the vehicle in the lead can be trusted by all other vehicles following it.

In [29], every agent has an individual trust vector established by a network information exchange. This is based on observations formed over neighbors found around an

agent, locally. The consolidation of local information exchange and the observations gives rise to trust-based formation control in a network. Consensus is reached by reflecting the influence of the neighboring agents.

Trust in networks such as sensors, wireless LANs, mobile ad-hoc, UAV and UGV teams can be augmented by setting up protocols such bilinear voting in [30]. It also discusses the methods of trust spreading in a network and the influence of local information on a particular agent. Behaviors in flocking is outlined to help analyze the rules of flocking and its application toward trust-based systems.

For transmission of secure information over wireless networks using sensors, [31] uses the approach of Kulback-Liebler (KL) distance to recognize and eliminate nodes or agents that have turned rogue. The system uses MAC (Media Access Control) addresses for verification purposes where each sensor has a key or password associated with it. Josang's belief model [32] and [33] is used to form opinions over the sensors during propagation of data over the wireless network. The opinions give the model a method to measure the trust in every sensor.

The RoboTrust model provides observations based on past history of the agents in [34]. Acceptance functions determine the favorability of an agent by the previously established observations. The output of the acceptance function is either 1 or 0 for being favorable or unfavorable accordingly. Agents that cannot be observed directly employ the technique of Indirect trust to determine the trustworthiness about other agents around them.

In [35], the model of trust is tested on a UAV by using visual data for feedback and human intervention when a system turns rogue. This case is similar to a supervisor-worker case where the performance of the UAV is closely monitored and when the control is lost, a human supervisor takes over. The human uses external commands to control the system. Trust in the system builds over time but when the system goes rogue, the trust value drops rapidly.

The concept of distributed control is utilized in [36] to achieve consensus. Decisions are made by each agent based on the local data available and a desired behavior is achieved accordingly. The local consensus is established by a voting protocol where the majority votes obtained by the neighbors is considered.

Auction based methods are used in [37] , where each agent in the team announces the time it is inclined to allocate in performing a particular task. A list of bids is maintained and it is constantly updated with the submitted bids. The list will be update the trust model adversely if the agent does not bid to perform a particular task. The trust model defines the bidding rules in the system. The agent that requires least amount of time to perform a task is chosen as the winner. Experiments were carried out on UAVs to verify their participation in teams based on bids.

In [38], graph theory is involved to describe the behavior of the network. It is assumed that nodes in a network trust themselves fully. The range of trust lies between -1 (distrust) and 1 (trust). '0' is a neutral zone where the trust model has no opinion over the nodes. A confidence parameter is defined by the weight function that plays a key role in deciding the level of trust. A higher confidence correlates to a high trust value of a particular node. Headers are used and they act as previously trusted nodes whose opinions have a higher leverage. Rogue nodes are eliminated from the network.

The definition of cooperation in networks is defined in [39] as a combination of competitive and altruistic trust. Competitive trust deals with selfish behavior in a team. In altruistic approach, the outcome is evaluated based on the perception of the team. Payoff functions were defined to describe the behavior of the team.

Chapter 6

Trust model for Robotic Manipulators

This section deals with an application of the trust model implementation on Robot manipulators. In multi-agent systems, the assumption is that all agents are equally trustworthy. In a real world, this assumption proves fatal and hence we seek to find a solution for the consensus problem in multi agent systems with the help of trust parameters. Trust algorithm in [34] is called RoboTrust, which is used to calculate trustworthiness in agents based on previous histories of the agent's interactions with its neighbors. In [30] , the agents use a consensus protocol to update their states. The RoboTrust model allows trust to be updated over a period of time by considering the histories of the agents. Acceptance functions use the observations made in the RoboTrust model and determines whether an agent is deemed favorable or not. Cultivation of trust in this process is suitable for our application.

The work on robot manipulators can be extended to be used on UAVs as well. The performance of a UAV can be based on factors such as velocity, acceleration, etc., and when a variation in one of these parameters in noted a threat prevention measure can be used to make a system more secure.

The next section deals with the utilization of uncertainties in Robot manipulators used for trust evaluation in a system.

6.1. Precision Considerations for Robotic Manipulators

Generally, the precision of a robotic system is represented by three metrics: resolution, repeatability, and accuracy (RRA).We redefine the traditional definitions for these RRA precision metrics to transform them into stochastic variables with mean and variance so that they can be used in making control decisions during assembly. The redefinitions are based on positioning results of controlled operation modes. Furthermore, we assume these distributions to be Gaussian.

The advantages in such approach are that the metrics are now dependent on available sensors, positioners, and control strategy and thereby can be used to improve performance for assembly cell with respect to required tasks. A more detailed discussion on the formulation of RRA rules can be found in [44].

The kinematics of a robotic manipulator, including the type of joints, actuators, and other geometric parameters significantly affects its precision (or positioning uncertainty) at the end-effector. The effect of parametric uncertainties in a serial robot chain on overall positioning uncertainty at the end-effector [45]. Two types of errors were considered: static errors due to misalignment and link parameter uncertainties, and dynamic errors due to inaccurate motion of individual links. Using uncertainty metrics the comparison of the precision of several different robot kinematic chain configurations is done. Assembly feasibility estimation was computed using the following analytical model [45].

$$ {}^{0}_{N}T(\theta) \cong \left[\prod_{i=1}^{n} \left[\left(e^{\xi_i \theta_i} + \delta\xi_i \theta_i \right) . e^{\xi_i \delta\theta_i} \right] \right] \left[{}^{0}_{N}T(\theta) \right] \qquad (1) $$

where

$\theta \rightarrow$ joint angle in case of revolute joints and displacement in case of prismatic joints;

$\xi \rightarrow$ twist vector representing an instantaneous link motion;

$T \rightarrow$ transformation matrix

In equation (1), the additive term is the "static error" or error due to link misalignment, whereas the multiplicative term is the "dynamic error" or error due to joint motion. In addition, the joints are also subjected to dynamic uncertainty which is a variable dependent on the range of motion executed by the joint. Thus, end-effector positioning becomes more and more uncertain with increasing distance along a robot path, and with repeated motions.

By a unique kinematic analysis, as described in [45] that quantitatively predicts the uncertainty in end-effector position due to individual motion errors in the manipulation hardware and their location in the overall kinematic chain.

6.2 Design of Trust Model for Robotic Manipulators

The RoboTrust model discussed in [34] makes the assumption that an agent has complete trust on its own state. In case of robotic manipulators this assumption is not practical because there are several sources of uncertainty as discussed in [45] that can affect an agent's trust on itself. The trust model for robotic manipulators is built from the uncertainty estimation model as given in equation (1) and [45]. The uncertainties in the robot manipulators are used to establish self-trust and global trust values.

shows a two arm Cartesian robotic manipulation system with each arm having 3 degrees of freedom i.e. x, y and z. The two arms are required to coordinate their motion in order to assemble a part *'P'*.

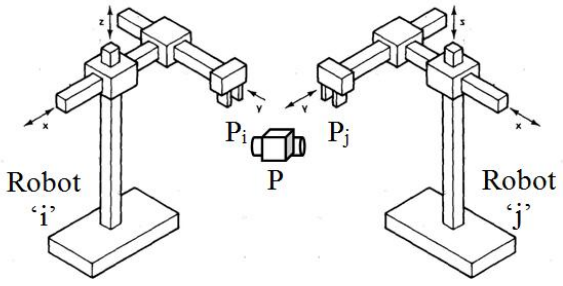

Figure 6-1 Two 3-DoF cartesian robotic manipulators coordinating to assemble a part 'P'.

step is to use the uncertainty estimation model, as given in equation (1) and [45], to calculate a self-trust value for individual robot arms. The second step involves building a global trust value between the two manipulator arms.

A discrete time Kalman consensus filter is used to calculate the trust estimates for specific tasks carried out by the manipulator arm pair. The filter uses the self-trust and global trust values. The Kalman filter uses the covariance of the measurement noise to obtain optimal results.

Calculation of self-trust

The robot pose is calculated as follows:

$$x_{i,d} = \begin{bmatrix} R & P \\ 0 & 1 \end{bmatrix} \tag{2}$$

where $i = 1, \ldots, N$ represents the number of manipulator arms, and d represents the desired case. With static and dynamic uncertainties, as given in equation (1), the actual poses for the manipulator arms are calculated by using the equation:

$${}^{0}_{N}\hat{x}_i(\theta_i) = \left[\prod_{l=1}^{n}\left[\left(e^{\hat{\xi}_{i,l}\theta_{i,l}} + \delta\hat{\xi}_{i,l}\theta_{i,l}\right)e^{\hat{\xi}_{i,l}\delta\theta_{i,l}}\right]\right]\left[{}^{0}_{N}\hat{x}_i(0)\right] \tag{3}$$

where $l = 1, \ldots, N$ are the links of individual manipulator arms.

Let $T_i(k)$ be the self-trust value generated by the uncertainties in the robot link due to pre-existing misalignment due to static and dynamic errors in Figure 6-2. The self-trust for each arm is calculated by using the equation:

$$T_i(k) = \exp\left(-\frac{\left\|\hat{x}_i - x_{i,d}\right\|}{\left\|x_{i,d}\right\| + \zeta}\right) \tag{4}$$

where

$\zeta \geq 0$ is the parameter designed to achieve a desired self-trust value.

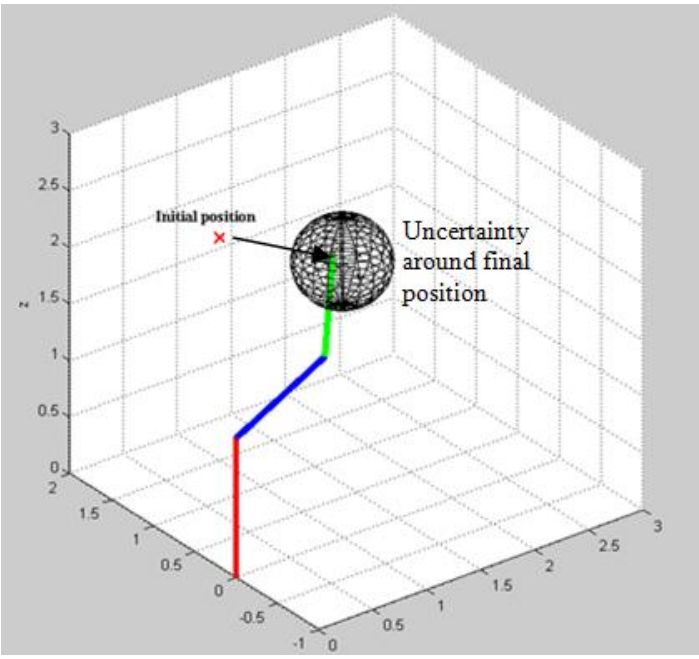

Figure 6-2 A 3-DoF Cartesian robotic manipulator with uncertainties in X, Y, Z.

Calculation of Global Trust

Standard graph theory notions are used in book [46]. A collection of N nodes are connected with each other using the directed graph interaction topology. $\mathcal{A} = [a_{ij}] \in \Re^{N \times N}$ is the associated weighted adjacency matrix of the directed graph.

The Update equation for the global trust between manipulator arms is given as follows:

$$T_{ij}(k+1) = T_{ij}(k) + \frac{1}{\sum_{m \in N_i, N_j} a_{im} a_{mj} + 1} \sum_{m \in N_i, N_j} a_{ij} a_{im} a_{mj} T_i(k) \left(\sqrt{T_{im}(k) T_{mj}(k)} - T_{ij}(k) \right) \quad (5)$$

where

$T_{ij}(k)$ → the previous global trust value;

d_i → the in-degree of node i;

$m \in N_i$ → is the neighbors of node i;

a_{ij} is the weight/adjacency matrix;

Confidence Factor

The number of agents under consideration is given as:

$$\sum_{m \in N_i} | T_{jm}(k) - T_{im}(k) | < \alpha \sigma_{ij}(k)$$

(6)

A new term $\sigma_{il}(k)$ is introduced here, which represents the confidence or reputation or accuracy factor. This decides if it falls in the range of other trustworthy agents in Figure 6-3. We could have a pre-defined range of values or calculate it using the normal distribution by calculating the mean and standard deviations. We can alter this value by changing α. This ensures that even though some values do not fall in the desired range but need not be eliminated completely while computing.

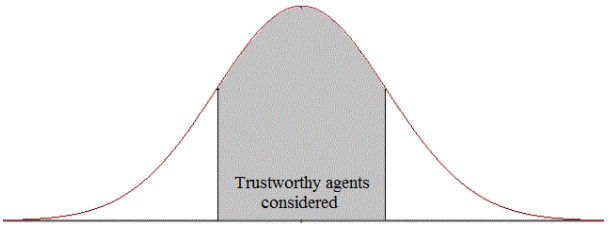

Figure 6-3 Range of trustworthy agents.

Fig.6-3. The overall trust of the model is decided based on computation of Global and self-trust as shown in the equations (4 and 5).

Multi-robot precision estimation based on the self-trust and the global trust models as discussed in the previous section is simulated in Matlab. Denavit-Hartenberg (D-H) model is used for modeling.

Case 1: Manipulation system with two 3-DoF robot:

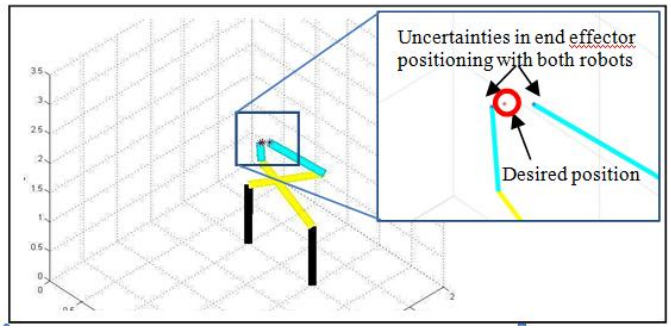

Figure 6-4 Simulated manipulation system with two 3-DoF robots plotted using Denavit-Hartenberg (D-H) model.

- the link misalignment between the degrees of freedom is within $\pm 5^{o}$
- the link lengths along each axes are considered to have an error within ±2 mm
- for each individual degrees of freedom, the maximum motion errors is taken as ±1% i.e. for every 1 mm motion range the stages are subjected to incur an error within ±0.01 mm.

6-4 shows one example case where, based on the above error conditions and using equation (1), the two robots try to reach a common goal position starting from their individual initial positions. The discrete time Kalman filter is used to track the uncertainties.

The tracking errors obtained from a Kalman filter for with and without trust parameter input is illustrated in Fig 6-5. The stability of the robot manipulator system increases with the use of the trust parameter. In the figure we notice some amount of noise present when the input to the filter is estimated without trust.

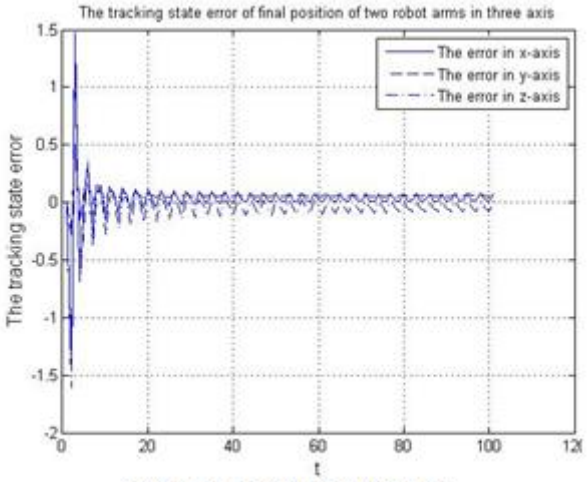

Figure 6-5 Tracking of robot uncertainties with the discrete time Kalman filter.

Figure 6-5(a) shows the uncertainty tracking without the self-trust model and Figure 6-5 (b) shows the uncertainty tracking with the self-trust model. As seen from the results the latter offers much stable performance.

Case 2: Extension of the Analytical Model to a 3-Robot System

The trust based uncertainty propagation and tracking using the analytical model is also validated for a 3-robot system as shown in Figure 6-6.

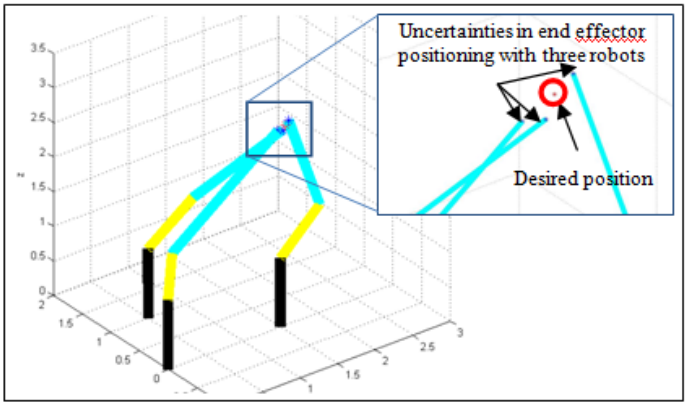

Figure 6-6 Cooperative manipulation task simulation using three 3-DoF robots.

The tracking of the robot uncertainties among the three robots using the discrete time Kalman filter is shown in Fig 6-7, 6-8, 6-9.

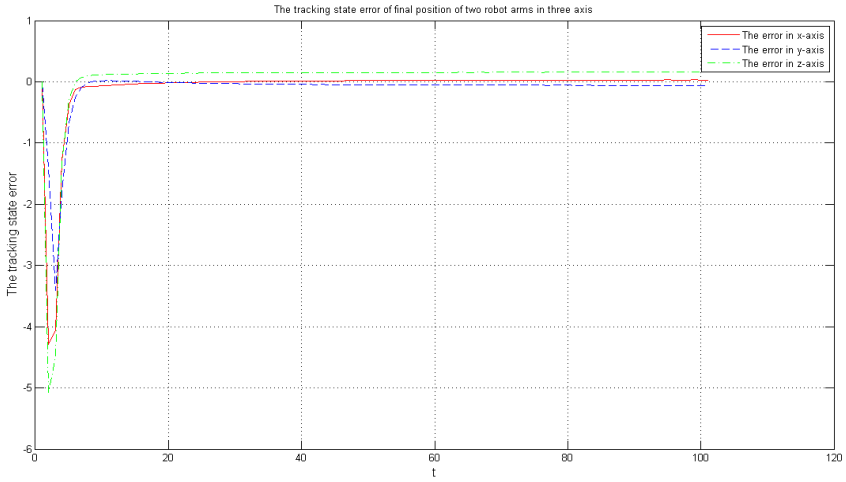

Figure 6-7 Tracking error between robot arm 1 and 2

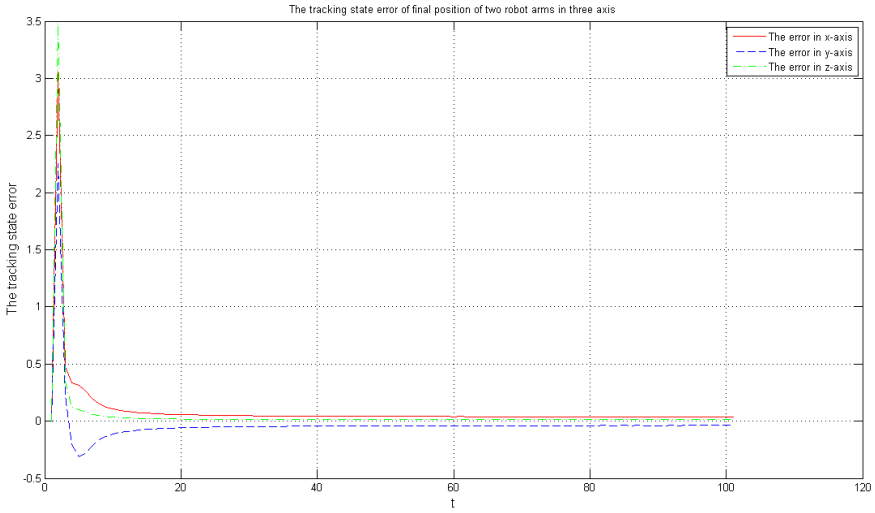

Figure 6-8 Tracking error between robot arm 2 and 3

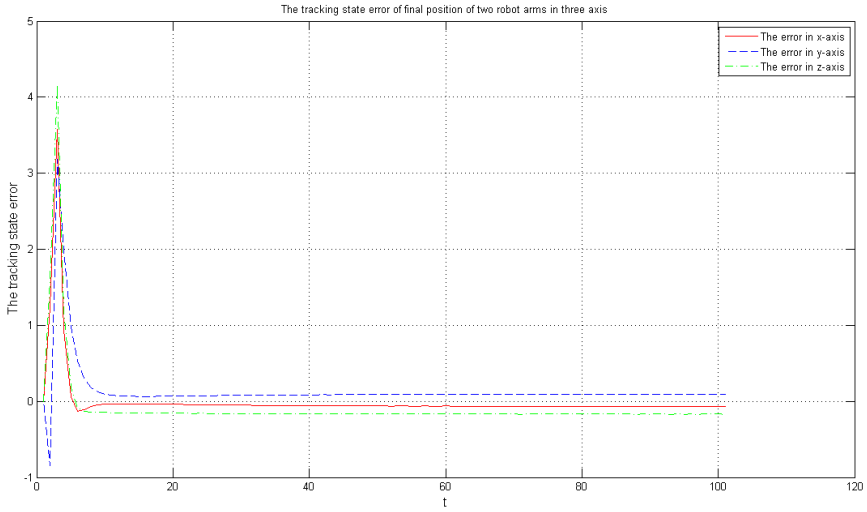

Figure 6-9 Tracking error between robot arm 3 and 1

The Robot Manipulator system is as shown in Fig 6-8 and Fig 6-9.

Figure 6-10 Modular manipulation test platform

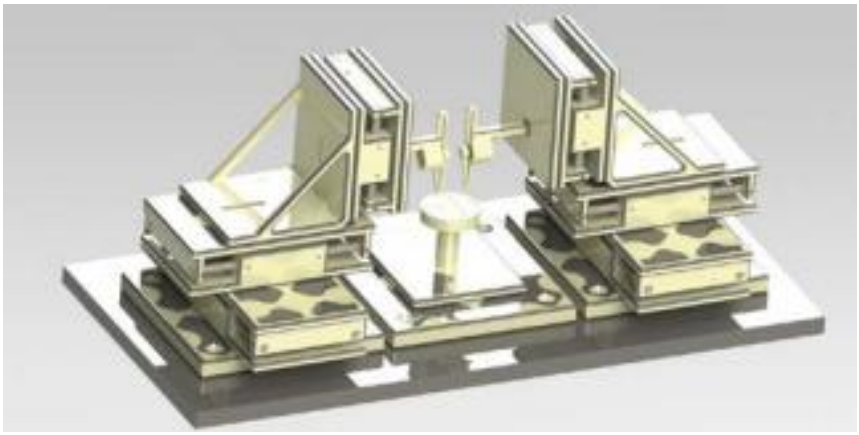

Figure 6-11 3D Rendering of the two robot manipulation system

Chapter 7

Conclusion

In summary, the hacking of the Parrot AR.Drone provides a clear example of similar systems which can be susceptible to de-authetication attacks. Exploitation of UAVs like the Parrot AR.Drone is a serious hazard to civilians. The AR.Drone is inexpensive and has the capability to transport hazardous payloads. It can be effortlessly infiltrated by a third party. Since the drone can be de-authenticated by another user, it helps us explore the issues in security of the device. The control of the drone can be switched easily from a valid client to a malicious client.

The use of UAVs is expanding rapidly and its need in private and public divisions has been in noticeable demand. In this thesis we have analyzed the importance of privacy linked to UAVs, so as to prevent exposure of sensitive and personal information. The different attacks and defense strategies are outlined as a caution for industries manufacturing UAVs. In the forthcoming years, UAVs will populate the sky and the most important necessity is to avoid incidents as a result of cyber attacks. The level of attacks will continue to rise and manufacturers as well as governments should make a collective attempt to safeguard these systems.

In the military, UAVs are being utilized for investigative purposes such as search and rescue missions. Areas where human intervention is impossible, UAVs can be used to monitor. The computing power of UAVs in the market today vary and have various features accordingly based on the platform . Information on board a UAV should be safeguarded from virulent attacks.

Interest in UAVs had been developing rapidly. Due to low-cost and light weight capabilities, they have been proven to be highly efficient and effective. Meticulous scrutiny reveals that UAVs are prone to issues in security. These challenges are unique and have to be recognized before a serious impact occurs.

Regulations on transport should be implemented to impart safety rules and training should be provided to users of UAVs. Security officers should be deployed to uphold the safety of

these devices and of civilians using them. Restrictions on usage should be informed to UAV manufacturers. Confidentiality of personal data should be maintained and privacy should be respected to avoid intrusions. Necessary laws should be passed for manufacturers and users to conform to these rules, as this will ensure safety.

The goal of this study was to investigate and analyze the flaws in security in UAVs and also to prevent such attacks. By testing on small scale UAVs, the results can be applied on larger drones. The solutions to defending such attacks were articulated. Various techniques such as Neural networks, DefCOMs and fuzzy logic based Intrusion detection systems were discussed. This helped gain exposure to vulnerabilities during cyber attacks.

In this thesis, a novel trust-modeling approach for cost and time efficient automation of multi-robot based cooperative manipulation systems was presented. The discussed technique takes a proactive approach, as opposed to the conventional reactive steps, to compensate for manipulation uncertainties. Derived from the manipulator kinematic configuration and precision specification of individual degrees of freedom, the discussed method computes self-trust values for each manipulator and subsequently a global trust model for the multi-manipulator system. This enables the implementation of an enhanced open loop control delivering rapid yet reliable performance. Simulation results for different multi-robot systems validate the efficiency and scalability of the trust-based cooperative automation technique.

This work can be extended to UAVs by using parameter estimation for performance variables such as velocity, acceleration, etc. and any variation in these parameters can be quantified for trust measurement. Once a threat is detected, it can be overcome using one of listed preventive techniques to help a rogue agent to be eliminated from a network of agents

Appendix A

Hacking the Parrot AR.Drone using Aircrack-ng and ROS

#Checking for any interferences
sudo airmon-ng check kill
#Monitor mode
sudo airmon-ng start wlan1
#Capture of packets and packet injection
sudo airodump-ng wlan1mon & sudo aireplay-ng -0 3 --ignore-negative-one -a <MAC address of drone> -c <Access point of the device used to control the drone> wlan1mon &
#Deauthentication complete
#Connect the drone to malicious client
sudo ifconfig wlan0 up
sudo iwconfig wlan0 essid "ardrone"
sudo iwconfig wlan0 ap any
sudo dhclient -v wlan0
#Control or land the drone using ROS commands
roslaunch ardrone_autonomy ardrone.launch
rostopic pub /ardrone/land std_msgs/Empty

Appendix B

Simulation of Trust for three Robot Arms

```
clc;
clear all;
close all;

  h = plot3(0,0,0);
  xlabel('x');
  ylabel('y');
  zlabel('z');
  p = get(h,'Parent');xlim(p,'manual');
  xlim(p,[0 3]);ylim(p,'manual');ylim(p,[-1 2]);zlim(p,'manual');
  zlim(p,[0 3.5]);
  axis vis3d;
  grid on;
  hold on;

  pause(0.01);

  In=eye(3);
  n=3;           %%%%% The number of point of each robot arm
  N=3;           %%%%% The number of robot arm
  T=100;          %%%%% Running time,50s
  Ts=1;          %%%%% Sample time
  Tc= T/Ts;      %%%%% The munber of iteration
  yita=0.1;

  s=zeros(n,N,n);  %%%%% The translation matrix
  s(:,1,1) = [0;0;0];
  s(:,1,2) = [1;0;0];
  s(:,1,3) = [1;1;0];
  s(:,2,1) = [0;0;0]; % from dN and rN
  s(:,2,2) = [1;0;0];
  s(:,2,3) = [1;1;0];
  s(:,3,1) = [0;0;0]; % from dN and rN
  s(:,3,2) = [1;0;0];
  s(:,3,3) = [1;1;0];

  sd1 = [0;0;0]; % from dN and rN
  sd2 = [0;1;0];
  sd3 = [1;0;0];

  theta=zeros(N,n);  %%%% The joint angle from previous x-axis to
current x-axis measured along previous z-axis
  theta(1,1) = 10;
  theta(1,2) = 20;
  theta(1,3) = 50;
```

```
  theta(2,1) = 20;
  theta(2,2) = 10;
  theta(2,3) = 25;

  theta(3,1) = 50;
  theta(3,2) = 00;
  theta(3,3) = 00;

  thetad1    = 25;
  thetad2    = 35;
  thetad3    = 60;

  alpha=zeros(N,n); %%%% The twist angle from previous z-axis to
current z-axis measured along x-axis
  alpha(1,1) = 30;
  alpha(1,2) = 30;
  alpha(1,3) = 60;

  alpha(2,1) = 70;
  alpha(2,2) = 40;
  alpha(2,3) = 30;

  alpha(3,1) = 20;
  alpha(3,2) = 30;
  alpha(3,3) = 50;

  alphad1    = 40;
  alphad2    = 40;
  alphad3    = 40;

  point=zeros(n,N,n);  %%%%%%% The actual position of the final
point of each robot arm
  pointd=zeros(n,n);   %%%%%%% The desired position of the final
point of each robot arm
  pointf=zeros(n,N);   %%%%%%% The initial position of the final
point of each robot arm
point(:,1,1)=[1.5;0;1];
  point(:,2,1)=[0.5;1;1];
  point(:,3,1)=[0;0;1];
  pointd(:,1)=[2;1;1];
  pointf(:,1)=[1.5;0;0];
  pointf(:,2)=[0.5;1;0];
  pointf(:,3)=[0;0;0];
```

```
Tself=[1;1;1];     %%%%% The self-trust----Initially, they were
assumed to be ones
  Tglobal=zeros(N,N,Tc);   %%%%%  The global-trust
  Tglobal(:,:,1)=[0 0.85 0.9; 0.8 0 0; 0 0.95 0]; %%%%%  The
initial values of global-trust
  Ag=[0 1 1; 1 0 0; 0 1 0];  %%%% The adjacency matrix---The
topology of the graph of robot arms


  G=eye(3); %%%%% The gain of the process noises

  Q=[0.003 0 0; 0 0.004 0; 0 0 0.004];  %%%%% The process noises
covariances
  P=zeros(n,n,N);   %%%%% The position error covariances
  P(:,:,1)=[10 0 0; 0 30 0; 0 0 20];
  P(:,:,2)=[20 0 0; 0 30 0; 0 0 10];
  P(:,:,3)=[10 0 0; 0 30 0; 0 0 10];

  g=[1,1,1]; %%%%  The pinning gain

  for tt=1:Tc
   for i=1:N
    for j=1:N
        temp1=0;
        for k=1:N
             if j==i
                temp1  = temp1;
             else
                 if k==i
                   temp1  = temp1;
                 else
                   temp1=temp1+Ag(i,k)*Ag(k,j);
                 end
             end
          end
          for k=1:N
%               if j==i
%                  Tglobal(i,j,tt+1)=1;
%               else
                  if k==i
                       Tglobal(i,j,tt+1)=Tglobal(i,j,tt);
                  else
Tglobal(i,j,tt+1)=Tglobal(i,j,tt)+(1/(temp1+1))*Ag(i,j)*Ag(i,k)*Ag
(k,j)*(sqrt(Tglobal(i,k,tt)*Tglobal(k,j,tt))-Tglobal(i,j,tt));
%%%%% Update equation of self-trust
                  end
```

```
end
            end
    end
  end

  for tt=1:Tc

      V1=[0.001*(0.5-rand(1));0.001*(0.5-rand(1));0.001*(0.5-
rand(1))];   %%%%% The noises in x-axis
      V2=[0.001*(0.5-rand(1));0.001*(0.5-rand(1));0.002*(0.5-
rand(1))];   %%%%% The noises in y-axis
      V3=[0.002*(0.5-rand(1));0.001*(0.5-rand(1));0.001*(0.5-
rand(1))];   %%%%% The noises in z-axis

   for i=1:N
    for j=1:N
       if j==i
         point(:,i,1) = point(:,i,1);
         point(:,i,2) = point(:,i,2);
         point(:,i,3) = point(:,i,3);
         P(:,:,i)     = P(:,:,i);
       else

        pointd(:,3) = [2.1;0.7;2.3];


      P(:,:,i)= inv(inv(P(:,:,i)+G*Q*G')+ inv(P(:,:,j)));
%%%% Update equation of error covariances
      point(:,i,1) =
returnUfromTheta(theta(i,1))*returnVfromAlpha(alpha(i,1))*s(:,
i,1) + point(:,i,1) + V1; % [0;0;1] is orig (previous)
position of point2
      point(:,i,2) =
returnUfromTheta(theta(i,1))*returnVfromAlpha(alpha(i,1))*retu
rnUfromTheta(theta(i,2))*returnVfromAlpha(alpha(i,2))*s(:,i,2)
+ point(:,i,1)+ V2;
      point(:,i,3) =
returnUfromTheta(theta(i,1))*returnVfromAlpha(alpha(i,1))*retu
rnUfromTheta(theta(i,2))*returnVfromAlpha(alpha(i,2))*...
returnUfromTheta(theta(i,3))*returnVfromAlpha(alpha(i,3))*s(:,
i,3) + point(:,i,2) +  P(:,:,i)*inv( P(:,:,j))*...
         Tglobal(i,j,tt)*(point(:,j,3)-point(:,i,3)) +
Tself(i)*P(:,:,i)*g(i)*(pointd(:,3)-point(:,i,3))+ V3;  %%%%
Update equations of positions

     Tself(i)=exp(-((norm(point(:,i,3)-
pointd(:,3)))/(norm(pointd(:,3)) + yita)));  %%%%% Update
equation of self-trust
```

```
pointx(i,tt+1)=point(1,i,3);
     pointy(i,tt+1)=point(2,i,3);
     pointz(i,tt+1)=point(3,i,3);
    end
   end

      pointdx(tt+1)=pointd(1,3);
      pointdy(tt+1)=pointd(2,3);
      pointdz(tt+1)=pointd(3,3);
      Ex(:,tt+1)=pointdx(tt+1)-pointx(:,tt+1);   %%%% The
tracking error between the desired position and the actual
position in x-axis
      Ey(:,tt+1)=pointdy(tt+1)-pointy(:,tt+1);   %%%% The
tracking error between the desired position and the actual
position in y-axis
      Ez(:,tt+1)=pointdz(tt+1)-pointz(:,tt+1);   %%%% The
tracking error between the desired position and the actual
position in z-axis
      Ecx12(tt+1)=pointx(1,tt+1)-pointx(2,tt+1); %%%% The
tracking error between the  position of the 1st arm and that of
the 2nd arm in x-axis
      Ecy12(tt+1)=pointy(1,tt+1)-pointy(2,tt+1); %%%% The
tracking error between the  position of the 1st arm and that of
the 2nd arm in y-axis
      Ecz12(tt+1)=pointz(1,tt+1)-pointz(2,tt+1); %%%% The
tracking error between the  position of the 1st arm and that of
the 2nd arm in z-axis
      Ecx23(tt+1)=pointx(2,tt+1)-pointx(3,tt+1); %%%% The
tracking error between the  position of the 2nd arm and that of
the 3rd arm in x-axis
      Ecy23(tt+1)=pointy(2,tt+1)-pointy(3,tt+1); %%%% The
tracking error between the  position of the 2nd arm and that of
the 3rd arm in y-axis
      Ecz23(tt+1)=pointz(2,tt+1)-pointz(3,tt+1); %%%% The
tracking error between the  position of the 2nd arm and that of
the 3rd arm in z-axis
      Ecx31(tt+1)=pointx(3,tt+1)-pointx(1,tt+1); %%%% The
tracking error between the  position of the 3rd arm and that of
the 1st arm in x-axis
      Ecy31(tt+1)=pointy(3,tt+1)-pointy(1,tt+1); %%%% The
tracking error between the  position of the 3rd arm and that of
the 1st arm in y-axis
      Ecz31(tt+1)=pointz(3,tt+1)-pointz(1,tt+1); %%%% The
tracking error between the  position of the 3rd arm and that of
the 1st arm in z-axis

  end

figure(1)
```

```
l2  = line([pointf(1,2),
point(1,2,1)],[pointf(2,2),point(2,2,1)],[pointf(3,2),point(3,2,1
)],'Color','k','LineWidth',10);
 l21 = line([point(1,2,1),
point(1,2,2)],[point(2,2,1),point(2,2,2)],[point(3,2,1),point(3,2
,2)],'Color','y','LineWidth',10);
 l22 = line([point(1,2,2),
point(1,2,3)],[point(2,2,2),point(2,2,3)],[point(3,2,2),point(3,2
,3)],'Color','c','LineWidth',10);

 l3  = line([pointf(1,3),
point(1,3,1)],[pointf(2,3),point(2,3,1)],[pointf(3,3),point(3,3,1
)],'Color','k','LineWidth',10);
 l31 = line([point(1,3,1),
point(1,3,2)],[point(2,3,1),point(2,3,2)],[point(3,3,1),point(3,3
,2)],'Color','y','LineWidth',10);
 l32 = line([point(1,3,2),
point(1,3,3)],[point(2,3,2),point(2,3,3)],[point(3,3,2),point(3,3
,3)],'Color','c','LineWidth',10);


set(l1,'ZData',[pointf(3,1),point(3,1,1)],'YData',[pointf(2,1),po
int(2,1,1)],'XData',[pointf(1,1), point(1,1,1)]);

set(l11,'ZData',[point(3,1,1),point(3,1,2)],'YData',[point(2,1,1)
,point(2,1,2)],'XData',[point(1,1,1), point(1,1,2)]);

set(l12,'ZData',[point(3,1,2),point(3,1,3)],'YData',[point(2,1,2)
,point(2,1,3)],'XData',[point(1,1,2), point(1,1,3)]);


set(l2,'ZData',[pointf(3,2),point(3,2,1)],'YData',[pointf(2,2),po
int(2,2,1)],'XData',[pointf(1,2), point(1,2,1)]);

set(l21,'ZData',[point(3,2,1),point(3,2,2)],'YData',[point(2,2,1)
,point(2,2,2)],'XData',[point(1,2,1), point(1,2,2)]);

set(l22,'ZData',[point(3,2,2),point(3,2,3)],'YData',[point(2,2,2)
,point(2,2,3)],'XData',[point(1,2,2), point(1,2,3)]);
set(l3,'ZData',[pointf(3,3),point(3,3,1)],'YData',[pointf(2,3),po
int(2,3,1)],'XData',[pointf(1,3), point(1,3,1)]);

set(l31,'ZData',[point(3,3,1),point(3,3,2)],'YData',[point(2,3,1)
,point(2,3,2)],'XData',[point(1,3,1), point(1,3,2)]);

set(l32,'ZData',[point(3,3,2),point(3,3,3)],'YData',[point(2,3,2)
,point(2,3,3)],'XData',[point(1,3,2), point(1,3,3)]);
```

```
hold on;

%
plot3(point(1,1,3),point(2,1,3),point(3,1,3),'*r','MarkerSize',
10);
%  hold on;
%
plot3(point(1,2,3),point(2,2,3),point(3,2,3),'*r','MarkerSize',
10);
%  hold on;

plot3(point(1,1,3),point(2,1,3),point(3,1,3),'*b','MarkerSize',
10);
 hold on;

plot3(point(1,2,3),point(2,2,3),point(3,2,3),'*b','MarkerSize',
10);
 hold on;

plot3(point(1,3,3),point(2,3,3),point(3,3,3),'*b','MarkerSize',
10);
 hold on;

plot3(pointd(1,3),pointd(2,3),pointd(3,3),'*r','MarkerSize',10)
;



figure(2)
plot(Ex(1,:),'r-');
hold on;
plot(Ex(2,:),'b-');
hold on;
plot(Ex(3,:),'g-');
grid on;
legend('The 1st agent','The 2nd agent','The 3rd agent');
xlabel('t','FontSize',12,'FontWeight','normal','Color','k');
ylabel('X','FontSize',12,'FontWeight','normal','Color','k');
title('The tracking state error of final position of robot arm
in x-axis');

figure(3)
plot(Ey(1,:),'r-');
hold on;
plot(Ey(2,:),'b-');
hold on;
plot(Ey(3,:),'g-');
grid on;
legend('The 1st agent','The 2nd agent','The 3rd agent');
xlabel('t','FontSize',12,'FontWeight','normal','Color','k');
```

```
legend('The 1st agent','The 2nd agent','The 3rd agent');
xlabel('t','FontSize',12,'FontWeight','normal','Color','k');
ylabel('Z','FontSize',12,'FontWeight','normal','Color','k');
title('The tracking state error of final position of robot
arm in z-axis');

figure(5)
plot(Ecx12(:),'r-');
hold on;
plot(Ecy12(:),'b--');
hold on;
plot(Ecz12(:),'g-.');
grid on;
legend('The error in x-axis','The error in y-axis','The
error in z-axis');
xlabel('t','FontSize',12,'FontWeight','normal','Color','k');
ylabel('The tracking state
error','FontSize',12,'FontWeight','normal','Color','k');
title('The tracking state error of final position of two
robot arms in three axis');

figure(6)
plot(Ecx23(:),'r-');
hold on;
plot(Ecy23(:),'b--');
hold on;
plot(Ecz23(:),'g-.');
grid on;
legend('The error in x-axis','The error in y-axis','The
error in z-axis');
xlabel('t','FontSize',12,'FontWeight','normal','Color','k');
ylabel('The tracking state
error','FontSize',12,'FontWeight','normal','Color','k');
title('The tracking state error of final position of two
robot arms in three axis');

figure(7)
plot(Ecx31(:),'r-');
hold on;
plot(Ecy31(:),'b--');
hold on;
plot(Ecz31(:),'g-.');
grid on;
legend('The error in x-axis','The error in y-axis','The
error in z-axis');
xlabel('t','FontSize',12,'FontWeight','normal','Color','k');
ylabel('The tracking state
error','FontSize',12,'FontWeight','normal','Color','k');
title('The tracking state error of final position of two
```

References

1. Wainner, Richard T., et al. "High altitude aerial natural gas leak detection system." *Final Report Prepared for US Department of Energy under Grant No. DE-FC26-04NT42268, PSI-1454/TR-2211* (2007).
2. McGonigle, A. J. S., et al. "Unmanned aerial vehicle measurements of volcanic carbon dioxide fluxes." *Geophysical research letters* 35.6 (2008).
3. Waharte, Sonia, and Niki Trigoni. "Supporting search and rescue operations with UAVs." *International Conference on Emerging Security Technologies (EST)*. 2010.
4. "Amazon Unveils Futuristic Plan: Delivery by Drone". *CBS News*. 1 December 2013.
5. "Triathlete injured by “hacked” camera drone" http://arstechnica.com/security/2014/04/triathlete-injured-by-hacked-camera-drone/
6. Hartmann, K., and C. Steup. "The vulnerability of UAVs to cyber attacks-An approach to the risk assessment." *Cyber Conflict (CyCon), 2013 5th International Conference on*. IEEE, 2013.
7. "Snoopy software can turn a drone is a data stealer". http://securityaffairs.co/wordpress/23374/hacking/snoopy-drone-data-stealer.html
8. " Howto Aircrack-ng (Simple Guide)". http://www.aircrack-ng.org/doku.php
9. "Getting Started with the Aircrack-Ng Suite of Wi-Fi Hacking Tools" http://null-byte.wonderhowto.com/how-to/hack-wi-fi-getting-started-with-aircrack-ng-suite-wi-fi-hacking-tools-0147893/
10. "Password Hacking with Wireshark" http://mile2.com/latest-news/password-hacking-with-wireshark.html
11. "What Are Man-in-the-Middle Attacks and How Can I Protect Myself From Them?" http://blog.trendmicro.com/what-are-man-in-the-middle-attacks-and-how-can-i-protect-myself-from-them/

12. Hazem (Moh'd Said) Hatamleh. "A Review and Comparing of all Hacking Techniques and Domain Name System Method"Contemporary Engineering Sciences, Vol. 5, 2012, no. 5, 239 - 250.
13. "What is a Trojan Virus?". http://usa.kaspersky.com/internet-security-center/threats/trojans.
14. Patrikakis, C.; Masikos, M.; Zouraraki, O. (December 2004). "Distributed Denial of Service Attacks". *The Internet Protocol Journal* 7 (4): 13–35.
15. Nemati, Hamid, ed. *Information Security and Ethics: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications*. IGI Global, 2007.
16. "How Encryption Works", computer.howstuffworks.com/encryption.htm
17. Bhuyan, Monowar H., et al. "Detecting distributed denial of service attacks: methods, tools and future directions." *The Computer Journal* (2013): bxt031.
18. Mirkovic, Jelena, et al. "Distributed defense against ddos attacks." *University of Delaware CIS Department Technical Report CIS-TR-2005* 2 (2005).
19. Abraham, Ajith, and Ravi Jain. "Soft computing models for network intrusion detection systems." *Classification and clustering for knowledge discovery*. Springer Berlin Heidelberg, 2005. 191-207.
20. E.Kesavulu Reddy. "Neural Networks for Intrusion Detection and Its Applications." Proceedings of the World Congress on Engineering 2013 Vol II, WCE 2013, July 3 - 5, 2013, London, U.K.
21. "What Is Fuzzy Logic?" http://www.mathworks.com/help/fuzzy/what-is-fuzzy-logic.html
22. Shanmugavadivu, R., and N. Nagarajan. "Network intrusion detection system using fuzzy logic." *Indian Journal of computer Science and Engineering*(2011).
23. Jiang, Tao, and John S. Baras. "Trust Evaluation in Anarchy: A Case Study on Autonomous Networks." *INFOCOM*. 2006.

24. Theodorakopoulos, George, and John S. Baras. "Malicious users in unstructured networks." *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*. IEEE, 2007.
25. Alborz Geramifard, Joshua Redding, Nicholas Roy, and Jonathan P. How. "UAV Cooperative Control with Stochastic Risk Models" American Control Conference (ACC), 2011.
26. "Trust Types". https://technet.microsoft.com/en-us/library/cc775736%28v=ws.10%29.aspx.
27. Mikulski, Dariusz G., et al. "Trust-based coalition formation in multi-agent systems." *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology* 11.1 (2014): 19-32.
28. Mikulski, Dariusz G., et al. "Trust dynamics in multi-agent coalition formation." *SPIE Defense, Security, and Sensing*. International Society for Optics and Photonics, 2011.
29. Haus, T., Palunko, I., Tolić, D., Bogdan, S., Lewis, F. L., & Mikulski, D. G. (2014). Trust-based self-organising network control. *IET Control Theory & Applications*, *8*(18), 2126-2135.
30. Lewis, Frank L. *Trust-Based Collaborative Control for Teams on Communication Networks*. TEXAS UNIV AT ARLINGTON, 2012.
31. Zhang, Wei, Sajal K. Das, and Yonghe Liu. "A trust based framework for secure data aggregation in wireless sensor networks." *Sensor and Ad Hoc Communications and Networks, 2006. SECON'06. 2006 3rd Annual IEEE Communications Society on*. Vol. 1. IEEE, 2006.
32. Jųsang, Audun, Roslan Ismail, and Colin Boyd. "A survey of trust and reputation systems for online service provision." *Decision Support Systems*43.2 (2007): 618-644.

33. Josang, Audun. "Trust-based decision making for electronic transactions."*Proceedings of the Fourth Nordic Workshop on Secure Computer Systems (NORDSEC'99)*. 1999.
34. Mikulski, Dariusz G., et al. "Trust method for multi-agent consensus." *SPIE Defense, Security, and Sensing*. International Society for Optics and Photonics, 2012.
35. Xu, Anqi, and Gregory Dudek. "Trust-driven interactive visual navigation for autonomous robots." *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012.
36. Baras, John S., and Pedram Hovareshti. "Effects of graph topology on performance of distributed algorithms for networked control and sensing."*Proceedings of the Workshop on Networked Distributed Systems for Intelligent Sensing and Control, Kalamata, Greece,(http://med. ee. nd. edu/)*. 2007.
37. Pippin, Charles E., and Henrik Christensen. "Cooperation based dynamic team formation in multi-agent auctions." *SPIE Defense, Security, and Sensing*. International Society for Optics and Photonics, 2012.
38. Jiang, Tao, and John S. Baras. "Graph algebraic interpretation of trust establishment in autonomic networks." *Preprint Wiley Journal of Networks*(2009).
39. Arney, David C., and Elisha Peterson. *Cooperation in social networks: communication, trust, and selflessness*. ARMY RESEARCH OFFICE RESEARCH TRIANGLE PARK NC, 2008.
40. Wilkinson, Glenn. "Digital Terrestrial Tracking: The Future of Surveillance."
41. "Aircrack-ng suite, Main documentation." http://www.aircrack-ng.org/documentation.html.
42. "SkyJack". https://github.com/samyk/skyjack.
43. "ROS Introduction" http://wiki.ros.org/ROS/Introduction.

44. D. O. Popa, R. Murthy and A. N. Das, "M³ - Deterministic, Multiscale, Multirobot Platform for Microsystems Packaging: Design and Quasi-Static Precision Evaluation," *IEEE Transactions on Automation Science and Engineering (T-ASE)*, vol. 6, no. 2, pp. 345-361, 2009.
45. A. N. Das and D. O. Popa, "Precision-Based Robot Kinematics Design for Microassembly Applications," in *ASME 2010 International Design Engineering Technical Conferences and Computers and Information in Engineering (IDETC/CIE) Conference*, Montreal, Canada, 2010.
46. F. L. Lewis, H. Zhang, K. H. Movric and A. Das, Cooperative control of multi agent system: optimal and adaptive design approaches, Berlin, Germany: Springer-Verlag, 2013.

Biographical Information

Chaitanya Rani is from Bangalore, India. She received her Bachelor in Engineering in Electronics and Communication Engineering from PESIT, Bangalore South, India in 2013. She earned her Master degree in Electrical Engineering from the University of Texas at Arlington and worked as a Graduate Research Assistant at the University of Texas at Arlington Research Institute. Her current research interests include Autonomous Vehicle Systems, Control Systems, Distributed Control Systems, Automation, Embedded firmware, Robotics, Consumer Electronics and Real time systems.