TRUST MANAGEMENT AND SECURITY IN SATELLITE TELECOMMAND PROCESSING

THESIS

Mark C. Duncan, Captain, USAF

AFIT/GCO/ENG/11-03

DEPARTMENT OF THE AIR FORCE
AIRUNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCO/ENG/11-03

TRUST MANAGEMENT AND SECURITY IN SATELLITE TELECOMMAND PROCESSING

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science

Mark C. Duncan, BSAE

Captain, USAF

March 2011

AFIT/GCO/ENG/11-03

TRUST MANAGEMENT AND SECURITY IN SATELLITE TELECOMMAND PROCESSING

Mark C. Duncan, BSAE

Captain, USAF

Approved:

Kenneth M. Hopkinson
Kenneth M. Hopkinson, PhD (Chairman)

14 MAR 11
Date

[illegible]
Lt Col Jeffrey W. Humphries (Member)

14 Mar 2011
Date

[illegible]
Maj Eric D. Trias (Member)

14 MAR 11
Date

Abstract

New standards and initiatives in satellite system architecture are moving the space industry to more open and efficient mission operations. Primarily, these standards allow multiple missions to share standard ground and space-based resources to reduce mission development and sustainment costs. With the benefits of these new concepts comes added risk associated with threats to the security of our critical space assets in a contested space and cyberspace domain. As one method to mitigate threats to space missions, this research develops, implements, and tests the Consolidated Trust Management System (CTMS) for satellite flight software.

The CTMS architecture was developed using design requirements and features of Trust Management Systems (TMS) presented in the field of distributed information systems. This research advances the state of the art with the CTMS by refining and consolidating existing TMS theory and applying it to satellite systems. The feasibility and performance of this new CTMS architecture is demonstrated with a realistic implementation in satellite flight software and testing in an emulated satellite system environment. The system is tested with known threat modeling techniques and a specific forgery attack abuse case of satellite telecommanding functions. The CTMS test results show the promise of this technique to enhance security in satellite flight software telecommand processing. With this work, a new class of satellite protection mechanisms is established, which addresses the complex security issues facing satellite operations today. This work also fills a critical shortfall in validated security mechanisms for implementation in both public and private sector satellite systems.

Acknowledgments

I would like to thank all of those who provided guidance, camaraderie and support during my time at AFIT. Your dedication and professionalism has enabled my success and will serve as a positive impression in my life. As new horizons dawn, I am fortunate to carry the legacy of my time spent at AFIT. Thank you.

Mark C. Duncan

Table of Contents

List of Figures

List of Tables

TRUST MANAGEMENT AND SECURITY IN SATELLITE TELECOMMAND PROCESSING

I. Introduction

1.1 Overview

This chapter gives a general introduction to the problem domain and provides an overview of the thesis research area. The significance of the problem being addressed and motivation for research is also presented in this section. Additionally, this chapter introduces the research goal and presents an overview of the thesis.

Satellite systems currently influence many aspects of modern society. From daily banking transactions to personal communications to global food production, modern society is dependent on the existence of satellite systems. These systems are launched and maintained by both commercial and governmental organizations and consist of ground and space-based segments. The primary entity in the ground segment of a satellite system is the satellite control center, or ground station, which commands and maintains the operational status of the satellite in orbit. The space-based segment of a satellite system consists of the orbiting satellite or constellation of satellites. Whether commercially or government owned and operated, satellite systems of all types have made their way into our lives.

Considering the vital role satellite systems play in modern society, these assets can be classified as national critical infrastructure. As defined in the United States Patriot

Act of 2001, critical infrastructure includes "systems and assets, whether physical or virtual, so vital to the United States that the incapacity or destruction of such systems and assets would have a debilitating impact on security, national economic security, national public health or safety, or any combination of those matters [1]."

Current public policy in the United States regarding critical infrastructure can be tied to Presidential Decision Directive 63 (PDD-63) "Critical Infrastructure Protection" published in May 1998 [2]. The guidance in PDD-63 describes critical infrastructure and sets national policy for protecting such infrastructure. This policy identifies responsibilities the Department of Defense (DoD) has for critical infrastructure protection, specifically, identifying the need to counter the threat of cyber attacks.

This added focus on cyber-based attacks stems from the increasing interdependence of critical national functions, such as banking, energy, and transportation, on advanced technology. One specific area of technology both supporting and serving as critical infrastructure is satellite systems. While the integration of new technology to these areas increases capacity and efficiency, it also makes the infrastructure more vulnerable to disruption and attack both physically and through cyberspace. Due to system integration, such an attack has the potential to create cascading failures. Because of the relatively recent progression from disconnected systems to complex interconnected systems, this threat is unprecedented, thus justifying research into new security mechanisms for critical infrastructure [2].

The DoD Critical Infrastructure Protection Plan (CIPP) published in November 1998, followed from PDD-63 to address the DoD plan to protect its portion of the nation's critical infrastructure. The DoD portion of national critical infrastructure is divided into sectors, one of which is "Defense Space" [3]. The Defense Space Infrastructure Sector detailed in the DoD CIPP consists of both space and ground-based assets. These include launch, specialized logistics, and control systems located worldwide on both DoD and commercially controlled sites [3].

These national and DoD policies state that satellite systems are currently considered a national critical infrastructure and require protection efforts. These efforts include vulnerability assessment and mitigation methods. This reality motivates research to identify potential vulnerabilities in satellite systems and to develop methods to mitigate associated risks.

To begin addressing security in satellite systems, this research considers common threats to space missions as presented in satellite security publications. The Consultative Committee for Space Data Systems (CCSDS) has published several reports concerning threats and security protocols relating to satellite systems [4,5,6]. A specific and the most significant threat discussed in these and related works involve the satellite command link, also known as the telecommand system. This threat is further categorized in this work. Furthermore, a specific forgery attack generally referred to as an abuse case is developed to illustrate satellite command link security issues.

Telecommand is a common term in the space industry referring to command and control communications through a wireless command link used to control satellites in orbit. According to the CCSDS report, *Telecommand: Summary of Concept and Service*, a telecommand system conveys control information from an originating source to a remotely located physical device or process [7]. In satellite systems, the controlled device is primarily a satellite bus, payload, or process aboard a spacecraft. The term telecommand used in this document refers to communications which initiate, modify, or terminate certain functions of a satellite [8].

Traditional space mission telecommand systems consist of centralized, mission-specific architecture [7]. These systems have unique components, data formats, and procedures for each space mission. This focus on mission unique technology and telecommand architecture is changing. The most significant of these changes are the development and implementation of open standards for spacecraft control. The primary motivation for the implementation of these standards is to reduce costs and satisfy cross support activities for satellite systems [5]. With the application of a general open telecommanding architecture, greater focus must be placed on satellite command link security.

Traditionally, logical security in satellite systems has not been a primary concern. Focus on system development in the space domain has primarily been on safe functionality rather than system security from malicious intent. The predominate reason for this is the critical fault intolerant nature of operating in space. The improper configuration of software onboard a satellite can leave a multimillion dollar space system useless, thus discouraging security features [9]. Another factor influencing this basic functional versus security focus is the threat profile in the space domain.

In the past, satellite systems were off limits to individuals and many organizations due to complexity and cost of entry. With the reduction of cost and the introduction of system standards, satellite system technology is now more widely available than ever [5]. In an effort to remedy the current Flight Software (FSW) security situation resulting from these factors, this work addresses the application of security features for satellite system commanding. The new features considered in this work utilize the concepts of trust management from the distributed information systems domain.

1.2 Preview

In summary, safety and security in satellite system commanding operations are in jeopardy given the increased access to the space domain, lack of space system security focus, and an increasing trend in global cyber threats [6,10]. It is proposed that satellite system safety and security can be improved with a proven trust management architecture which addresses common cyber threats to space assets. The focus of this research is to develop an efficient and effective method of applying interaction trust via a Trust Management System (TMS) to satellite system commanding for enhanced mission safety and security.

Chapter II presents a review of related literature that introduces the satellite system domain in detail along with security and trust management principles. Chapter III details the proposed TMS for satellite telecommanding incorporating multiple trust mechanisms and introduces abuse case methodology for evaluating the system. Chapter IV presents the experiment setup used for testing FSW and the developed TMS with a forgery attack abuse case. Chapter IV also covers results of the TMS testing and performance characteristics of the TMS. Chapter V provides an analysis of the results with recommendations for further research in the field.

II. Literature Review

2.1 Introduction

This chapter presents a review of literature, concepts and existing work relevant to the development of a trust-based security model for satellite systems. First, a review of the space domain as it relates to satellite systems is presented, covering the roles, missions, operating environment, and components of satellite systems. Second, literature related to satellite system threats is introduced. Third, security principles and threats which apply to satellite and computer systems are reviewed. Finally, the concept of trust, as it relates to distributed information systems and trust related work is presented.

This thesis focuses on threat detection and mitigation in satellite system telecommanding operations through the application of a Trust Management System (TMS) to satellite Flight Software (FSW). Most threats to spacecraft telecommand links are a result of their Radio Frequency (RF) transmissions being broadcast through an open medium [5,6]. To understand the threats and risks to satellite telecommanding systems an investigation of existing work which classifies threats to satellite systems is performed. To form a response to these threats, concepts of computer and satellite security are reviewed. Principals of trust management in distributed information systems and other fields are also examined to support this thesis. Concepts from relevant trust research are utilized to realize the primary goal of this work, to develop and apply a new satellite security mechanism utilizing trust management theory. This effort requires a general understanding of space system missions, space environment, and system components. These principles of satellite systems lay the foundation for common satellite operations upon which system security can be evaluated.

2.2 Satellite Systems Role

Satellite systems influence many aspects of modern society. From daily banking transactions to personal communications to global food production, our society is dependent on the existence and operation of satellite systems.

Not only do many commercial industries rely on the use of satellite resources to conduct daily operations, government entities have also grown similarly dependent on satellite systems [11,12]. The U.S. DoD currently utilizes many types of satellite systems and has contributed to the advancement of satellite systems in general. Official policies of the U.S. government emphasize the critical nature of these systems to the prosperity of the nation and call for the protection of these systems [13,12,14].

Traditionally, satellite systems operating in the public and private sectors have been large-scale and highly proprietary in nature. With more than 50 years since the launch of the first satellite, much has changed in the way of satellite systems and the computer components vital to their operation. Satellite system components are now becoming more standard across missions along with the integration of mission operations with internet technology. This changing culture in satellite system development and operations will increase efficiency but also provide new security challenges [6]. One example of this phenomenon is illustrated by the CubeSat standard and its various derivative projects. This new trend in satellite systems does not only apply to small satellites, however applies to all of the mission areas for satellite systems.

2.3 Satellite Systems Mission Areas

The missions of satellite systems can be categorized into four main areas: Remote Sensing; Communications; Precision Navigation and Timing; and Science/Exploration [15]. One key aspect of satellite systems is that the primary mission will determine in which orbit the satellite will operate [11,16,17,15], see Section 2.4.1. The following sections examine the characteristics of each mission area.

2.3.1 Remote Sensing.

The primary mission of remote sensing satellite systems is to observe the Earth and other objects from orbit. Satellites in orbit provide a unique observation platform from which to view the Earth. Remote sensing satellites utilize Radio Detection and Ranging (RADAR), optical, and other sensors to collect images and measurements. These systems can also detect Infrared (IR) and other emanations from Earth.

Optical observation is used to produce images of the Earth's surface which can be analyzed to provide valuable information. This information aids military operations and intelligence, as well as documentation of urban development, scientific research, and predictions of crop yields for global food supply planning [18]. Additionally, IR sensors aboard military satellites are used to provide missile launch and nuclear detonation detection [19]. One of the earliest benefits of space systems was the prediction and analysis of Earth's weather patterns. Optical and IR sensors are used to observe visible weather phenomena as well as collect atmospheric measurements for weather forecasting [20]. These satellites are even used to monitor space weather, see Section 2.4.2.

2.3.2 Communications.

Communication satellites revolutionized the way humans communicate. A satellites unique field of view makes it an ideal communications relay platform. Satellite communications (SATCOM) links have been used to provide intercontinental communications for major telecommunications providers since the mid 1960s [16].

Large corporations use SATCOM to link together many geographically separated organizational units into one network in order to facilitate logistics and enhance daily operations [11]. SATCOM is also extensively used by military forces to conduct operations around the globe.

2.3.3 Precision Navigation and Timing.

The United States government pioneered the development of precision navigation and timing satellite constellations beginning with the TRANSIT satellite constellation in 1960 and culminating with the NAVSTAR Global Positioning System (GPS) which became operational in 1993 [15]. NAVSTAR GPS satellites, along with ground control and monitoring stations, provide accurate three-dimensional locations and timing for use in civilian and military operations worldwide. After the introduction of GPS for government use, the tremendous commercial implications for the GPS signal have led to its widespread civilian use and eventual dependence.

Examples of private sector and individual use of GPS include banking, transportation and communications. The banking and global financial markets now rely on GPS timing to synchronize transaction systems. Individuals routinely use GPS signal for navigation assistance. Finally, as the wireless telephone market developed with the freely available timing provided by GPS, these systems are now dependent on this method of time synchronization [21].

2.3.4 Science/Exploration

Satellites orbiting with science/exploration missions continue to advance technology used in satellite systems and contribute to many diverse fields of science. Such missions have led to advancements in Earth and materials science, as well as astronomy. These science/exploration missions provide immeasurable value to the human understanding and way of life through invaluable advancements and discoveries [22].

Examples of these space related technologies being transferred to two diverse areas of our daily lives are advancements in medical imaging and the development of cordless tools. The medical imaging technology advancement stems from the charged coupled devices (CCDs) used aboard the Hubble space telescope which convert light directly into digital images. These devices are now used to image the human body and differentiate between benign and cancerous tissues. Additionally, technology developed to recover lunar samples was directly transferred to domestic use as battery powered power tools and appliances [23].

2.4 Space Orbits and Environment

Operating in space presents many challenges to the success of satellite systems. In order to conduct a successful satellite mission, considerations must be made with regard to a satellites orbit, and the space environment. The space environment consists of extreme temperatures and pressure, along with multiple forms of radiation. Additionally, these environmental factors are impacted by the orbit at which a satellite is operating [17].

2.4.1 Orbits.

A satellite's primary mission will dictate the orbit in which the satellite will be placed. The orbit placement in turn will determine when and how long a ground station has a satellite within field of view, and how much transmission power is required for communications between the satellite and ground stations. Orbit placement also influences satellite power considerations, stability, and attitude control options [15]. There are two satellite orbit shapes: circular and elliptical. For both orbital shapes, the center of the Earth is in the plane of the orbit with an inclination taken between the orbital and equatorial plane. Circular orbits are further subcategorized by altitude: Low Earth Orbit (LEO), Medium Earth Orbit (MEO) and geosynchronous orbits [16,15,24].

LEO satellites operate with an altitude up to 1500 km and are primarily used for remote sensing and science missions, with some communication applications being made with advanced cross linking satellite platforms [15]. The communication satellites operating in LEO are primarily used to provide global mobile telephone service. Advantages and disadvantages for LEO satellites are used to determine the orbits suitability for a particular satellite mission. The advantages of LEO are [17,15]:

- reduced cost to place payloads in orbit
- reduced communication transmission power requirements
- simplified satellite attitude control
- lower latency for communications
- higher resolution for remote sensing applications

Some disadvantages for LEO orbits are [17,15]:

- short ground station access periods
- high satellite velocities complicate communications
- short orbital lifespan due to increased drag at lower altitudes

MEO satellites operate at an altitude of around 20,000 km, which is well outside the Earth's atmosphere. The increased altitude relative to LEO gives MEO satellites a broader view of the Earth, making them more suitable for applications requiring more global coverage. MEO class orbits can be semi-synchronous with the Earth's rotation with a satellite making two revolutions around the Earth in one day. These orbits are most suitable to modern precision navigation and timing applications such as NAVSTAR GPS [15].

Geosynchronous satellites operate at an altitude of 35,780 km and have the special property of orbiting the Earth once every 24 hours [15]. This allows the spacecraft to remain in a semi-fixed position relative to a specific ground station. Geosynchronous satellites will have some perturbation in location relative to a fixed spot on Earth, which may require ground stations communicating with them to track this slight movement. Geostationary Earth Orbit (GEO) is a geosynchronous orbit which has an inclination of exactly zero degrees. GEO satellites will remain over the same fixed spot on Earth with proper station keeping [15].

GEO and geosynchronous satellites are ideal for communication missions where the satellite remains fixed in the sky and can reliably relay communications between two ground stations within the satellites field of view. Each geosynchronous satellite has a field of view of approximately one half of the Earth. Global communications, with exception of the high latitudes, are achievable with a three satellite constellation [15].

Elliptical orbits have unique properties which can be useful for communication missions to high latitude areas. These orbits fill the gaps in coverage left by GEO satellites, which at best reach latitudes of 81 degrees. One specific type of elliptical orbit which can be used to simulate a stationary satellite in GEO is the Molniya orbit. These elliptical orbits with high eccentricity and an inclination of 63.4 degrees provide long dwell times over a location on Earth. A constellation of three Molniya satellites are required to provide continuous service to a single ground location [15].

2.4.2 Space Environment.

The space environment consists of a vacuum with varying levels of cosmic and solar radiation. Satellites in orbit do not have the full protection from radiation provided by the Earth's magnetosphere. This leaves satellites vulnerable to space weather conditions which has a detrimental effect on critical systems even when protection mechanisms are in place [17].

Space weather consists of magnetic fields, charged particles, and radiation. The dominant factor in space weather is solar wind from the sun consisting of charged particles and radiation bursts. The effects of solar winds vary over time, as their strength fluctuates and as they impact Earth's magnetic field. Highly charged particles in the space environment, from solar wind, often disrupt electronics aboard satellites [17].

The most common occurrence of electronic disruption or failure aboard satellites is the Single Event Upset (SEU). A SEU occurs when an ion or electro-magnetic radiation interferes with an electronic circuit in such a way that information stored in the circuit as bits are corrupted. This action often results in a failure of the satellite's onboard computer logic. These errors are generally not fatal for the spacecraft and normal operation is typically resumed after resetting the system [25]. Certain orbits and areas in space are more prone to these types of events. One commonly known area in which satellites experience an increased likelihood of a SEU is called the South Atlantic Anomaly (SAA).

The SAA is an area over Brazil and the south Atlantic ocean where space borne radiation comes closer to the Earth than any other place. This area is caused by a dip in the Earth's magnetic field allowing cosmic rays and charged particles to reach lower altitudes. Satellites crossing this area are exposed to higher levels of radiation, which results in the increased chance of a SEU [26].

These unpredictable, disruptive events have an impact on system design. As a result, engineers have designed systems with failsafe defaults and recovery modes which in turn increase mission safety. These actions lead to systems which are more mission safe and in some cases increases system complexity.

Mission safety is a concept in which the satellite system is robust to failure and the likelihood of satellite operators losing control due to human error or space weather is reduced. The disadvantage of a sole focus on mission safety is that systems may be more vulnerable to malicious actions, resulting in reduced system security. Though the concepts of mission safety and system security appear to be competing design principals, a balance between the two must be achieved which serves the overall requirements for the mission.

2.5 Satellite System Components

The components of satellite systems are generally similar, however, the mission determines which components are found in each system. Components are organized by their location in the system, which is divided into three categories: ground segment, space segment, and subscriber segment. Not all satellite systems require all segments of the general satellite system [16].

The satellite system can be broken down and described by functional areas consisting of major components. The exact configuration of these functional areas are influenced by and provide support to the system's primary mission. The satellite system functional areas described here are Satellite Hardware, Satellite Software (programmed logic), Ground Station Hardware, Ground Station Software (programmed logic), and Telecommand Architecture [16,15].

2.5.1 Satellite Hardware.

The orbiting hardware, or satellite, typically contains the following subsystems: Telemetry Tracking and Command Subsystem (TT&C), Electrical Power System (EPS), Propulsion System, Attitude Determination and Control System (ADACS), and Thermal Control System (TCS). These subsystems constitute the satellite main bus. The main bus is constructed to support the primary payload. The following subsections will describe the functions of the main bus subsystems and the payload integration [16,15].

2.5.1.1 Telemetry Tracking and Command (TT&C) Subsystem.

The TT&C subsystem hardware consists of a flight computer, radio, and antenna for communicating with a controlling ground station. The TT&C subsystem provides satellite status data (telemetry) along with functions to command the satellite and control the other subsystems. The TT&C subsystem is also used to provide a tracking and ranging service to the ground station. Tracking and ranging data is used to accurately point high gain antennas towards the satellite and to provide accurate orbit determination for the satellite's mission [16,15].

Commands from ground stations are received by the TT&C subsystem and command specific actions are executed. These commands can be intended for the TT&C subsystem itself or for any of the other subsystems. Additionally, status information from all of the satellite subsystems is collected and formatted for transmission as satellite telemetry to listening ground stations.

2.5.1.2 Electrical Power System (EPS).

The spacecraft's payload and satellite support systems (bus) require power provided by the EPS to operate. The EPS manages power generation, storage, and distribution throughout the spacecraft. These actions are done in conjunction with the logic present in the satellite's flight computer. The power subsystem hardware consists of an EPS controller, power source, and batteries [16].

The distribution of a spacecraft's power is managed by the EPS controller. The EPS controller continuously monitors and adjusts connections to the power sources and the charge/discharge rates of the onboard batteries. Additionally, the EPS controller switches power to each of the spacecraft bus systems and payload as necessary, while reporting power status information to the flight computer [27].

The primary power source for most satellites is photovoltaic solar cell arrays. The power output of a solar array is proportional to the angle of incidence of solar rays on the panel. If the panel is misaligned with the Sun's radiation such that the angle of incidence is zero, then no power will be produced [16].

Most satellites will periodically encounter eclipse periods in which no direct sunlight reaches the spacecraft. The frequency and duration of these eclipse periods are determined by the orbit of the satellite. During these periods, the orbiting spacecraft must rely on a power source other than solar. Battery systems fill the gaps in power supply for satellites during eclipse periods [15].

Satellite commands which orient a satellites solar cell arrays or configure the EPS for power distribution are critical to the successful operation of a satellite. Any malicious or otherwise improper processing of commands which modify the satellites power configuration has the potential to leave the satellite crippled or otherwise inoperative. This is one example of where specific satellite telecommands present a potential vulnerability to satellite systems.

2.5.1.3 Propulsion subsystem.

Most long term satellites require propulsion systems to make changes to their trajectory after being placed into orbit. These trajectory changes may be significant or minor. Significant changes in orbit are generally made during the initial phase of a satellites operation to reach the desired mission orbit. Once the mission orbit is established, an onboard propulsion system is needed to maintain the orbit [15].

2.5.1.4 Attitude Determination and Control subsystem (ADACS).

The ADACS in a spacecraft is used to maintain sensor and antenna orientation for pointing. The ADACS is composed of control logic, sensors, and mechanical systems for adjusting the attitude of the satellite. The control logic for the ADACS is maintained by the spacecraft flight computer or by a secondary microcontroller. The ADACS can be dynamically adjusted via the satellite command link. There are many sensors and techniques for satellite attitude determination, including star tracking, sun tracking, IR Earth sensing, and RF tracking [15].

With accurate spacecraft attitude information, the flight computer can make adjustments in position with the attitude control systems. These systems are used to accurately point sensors at a desired location or place the spacecraft in a desired orientation [15]. Any failure or malicious activity disrupting the operation of these systems could result in total loss of the satellite.

2.5.1.5 Thermal Control System (TCS).

Space is extreme with respect to temperature. External satellite components can experience temperatures ranging from -200 to +150 degrees Celsius [15]. These temperature extremes drive the development of spacecraft thermal control systems. Two methods of thermal control are passive design techniques and active thermal control systems. Passive design techniques rely on material conduction and radiation properties. Active thermal control systems include electric heaters and radiators which must be operated as conditions change on the satellite. A satellite's thermal design is critical to its successful operation. Thermal systems are monitored by telemetry sent through the command link. Adjustments to active components of the thermal control system can be made via specific thermal control commands [15].

2.5.1.6 Payload.

The main system payload will correspond to the satellite's mission. The payload can consist of sensors, communication devices, or scientific equipment. The deployment and control of the payload is facilitated by the flight computer and command system with support from the other bus systems for power, pointing, and thermal management [15].

2.5.2 Satellite Flight Software (FSW).

The satellite FSW running on the main flight computer is a critical component of the satellite and manages all aspects of the satellite hardware (TT&C, EPS, Propulsion, ADACS, TCS, Payload) [28]. Portions of the FSW must be specifically tailored to each function of the spacecraft bus. Some of the spacecraft management can be automated to handle faults as they occur and alleviate the need for intervention by the commanding ground station.

2.5.3 Ground Station Hardware.

Satellite systems are managed by ground stations which send control information to the orbiting satellite in the form of discrete commands. These control messages are commands, which specifically address each of the satellite subsystems previously mentioned (TT&C, EPS, Propulsion, ADACS, TCS, Payload). Additionally, the ground station receives satellite telemetry of the satellite's status [16].

A typical satellite ground station has three main components: RF Interface, Signal Processing, Mission Execution. The RF Interface is composed of the antenna, low noise amplifier, power amplifier, signal down converter and signal up converter. The Signal Processing component consists of a transceiver, Terminal Node Controller (TNC), and modem. The mission execution segment typically consists of command terminals operating ground control software and integrates the mission specific processing components with the other ground station components. Satellite systems may utilize multiple ground stations for redundancy purposes [16].

2.5.4 Ground Control Software.

Ground control software facilitates the commanding of satellites within a satellite system. The software runs on computers at the ground station or on remote systems which connect via networks to the ground station. The function of ground control software is to transmit commands to the satellite, receive and interpret telemetry data passed back from the satellite, and pass updates to the satellite software [16].

The design and implementation of ground control software is driven by mission requirements. The architecture of ground control software can be divided into two sections: common ground control components and mission unique components. The common ground control components are not mission specific and can be used with multiple satellite systems. An example of software which has been developed to serve the function of common ground control components is the Common Ground Architecture (CGA) command software. CGA is ground control software with a large percentage of the architecture consisting of reusable ground control code. This modular design allows CGA to support multiple satellite missions with the same core components and mission specific applications. Each satellite system CGA supports has specially designed mission unique components [29].

2.5.5 Telecommanding Architecture.

Satellite system telecommanding architecture will differ by implementation, however some key concepts are presented as features contributing to the safety and security of satellite commanding in general. Additionally, some examples of satellite telecommanding architecture are covered to illustrate the concept and highlight areas where trust management concepts may be applied.

The telecommanding architecture and implementation play a crucial role in the safety and security of a satellite system due to the open nature of communications between ground stations and satellites. Critical vulnerabilities in the telecommanding architecture implementation can allow events, whether malicious, accidental, or environmental, to disrupt or destroy a satellite [6]. The following features have been seen in satellite telecommanding architectures and are similar to concepts found in distributed information systems:

- Satellite addressing is implemented in a telecommanding architecture to direct commands to a specific satellite or decoder. This feature affords some protection from commands being erroneously processed by the wrong satellite or satellite subsystem [15].
- Command register verification confirms commands to be processed aboard a satellite by transmitting the command queue to the ground station for acknowledgment before execution [15].
- Encryption is used to provide the security service of confidentiality to the telecommanding architecture. It can be implemented in many levels of the architecture resulting in varying impacts on system performance, complexity, reliability, and security [5,30].
- Command counters in the telecommanding architecture provide an element of security and safety to the system. The command counter assigns a unique number to each command being transmitted to the satellite. Command counters are primarily implemented to ensure commands are executed at most once and in the proper order. If commands are processed out of order or in

duplicate, the system could become unstable resulting in mission degradation or loss of the satellite [15].

- Authenticated commands in the telecommand architecture is implemented by checking the command counter or through more complex cryptographic mechanisms [15]. The use of a command counter for command authentication provides command execution safety, where cryptographic user authentication enables system security.
- The use of time stamps in telecommanding architecture involves assigning a time stamp to each telecommand message. The time stamp is used by the satellite to verify the sequence of commands being received. Time stamps feature is also useful in detecting the replay of a command [15].

The following are examples of how the telecommand architecture can be implemented in a satellite system:

The command execution of a satellite as described by Patton [15] is typically a two-step process. First, the satellite operator selects a command which is formatted by the ground control software and equipment for transmission to the satellite. This command formatting appends a preamble to the transmitted command. The preamble contains an address key which identifies the particular satellite for which the command is intended. This specific address key provides protection against stray signals being received and executed by the satellites TT&C subsystem. Once the TT&C subsystem receives a command from the ground station with the proper address, the contents of the command register (current command to execute) are transmitted to the ground station for verification. If the command is successfully validated by the ground station, the transmission is acknowledged by transmission back to the satellite, at which time the command is executed. It should be noted that the commanding procedures for satellite systems can vary greatly between systems of differing size, complexity, and purpose [15].

The command execution sequence for the satellite system used as a model in this research is a derivative of CubeSat FSW and concept of operations. In this model, commands are formatted at the ground station with the ground control software CGA. CGA interprets human readable commands and builds a data stream which is then transmitted to the satellite. The data stream transmitted to the satellite contains ground station and space station ID numbers. These are used to confirm that the command is originating from a valid ground station ID and is directed to the proper satellite interface. The command data stream also contains a message protocol and command identifier which are used to determine the satellites response to the command. As a safety feature, a command authentication count is included in the message header. The command authentication count is used in conjunction with most commands and verifies the satellite and ground station is in sync during the commanding process. Command arguments (parameters) are marshaled after the authentication count. CGA computes a checksum over the command header and arguments, and appends it to the end of the command data stream. The checksum ensures data integrity during the command transmission [29,31].

2.6 Space System Threats and Security

This section describes space system threats and corresponding security mechanisms. However, this treatment of space system threats is not exhaustive, but is provided to serve as a broad overview and background for the development of scenarios to aide in the testing of new security features for satellite telecommanding. Information regarding possible threats to space missions is necessary for mission planners to better understand the security mechanisms and policies required to mitigate them. All systems are subject to threats which may result in the loss of data or catastrophic damage to the system [6].

A threat is a potential violation of security. The occurrence of a violation is not necessary for a threat to be present. An attack refers to activity related to the violation of security. In order for a successful attack to take place, the system must be vulnerable to the threat in action. The existence of a possible violation of security requires actions to be taken which guard against threatening activity and mitigate system vulnerabilities [32].

The following sub-sections serve as an overview of the most common threats to space missions. This information highlights the findings presented in the CCSDS report *Security Threats Against Space Missions* [6].

2.6.1 Data Corruption.

Data corruption occurs as the result of a fault in either the ground or space segment of a satellite system or by the intentional or unintentional action of an individual. This corruption event may take place in the hardware or software of the satellite system's components. Common faults include hardware failures or a SEU in the spacecraft. The effects of data corruption can range from an unnoticed anomaly in telemetry data to catastrophic loss of the spacecraft due to the processing of a corrupt command [6].

2.6.2 Interception Of Data.

Data communications with spacecraft are achieved via RF signals which are subject to interception. The extent to which this threat applies to a space mission is dependent upon the orbit in which the space segment of the system is operating. LEO missions are less susceptible due to the short access period and small beam width of the downlink signal. Conversely, missions operating in high orbits such as GEO have large downlink beams and long access periods and therefore increased susceptibility to interception. Transmissions from ground stations are typically less susceptible to interception due to the highly directional antennas and small beam widths used to communicate with satellites. Signals may be intercepted by listening ground stations and by signal intelligence gathering aircraft or spacecraft [6].

2.6.3 Jamming.

Persistent RF interference is characterized as jamming. The RF signals used for communications with spacecraft are susceptible to interference. Interfering signals can be intentional or unintentional and can result in link loss or denial of communications with the satellite. This interference is accomplished by transmitting a competing signal on the same frequency the satellite is operating. Interference can originate from a ground station or from a third party satellite orbiting within line of site of the mission ground and primary satellite [6].

2.6.4 Masquerade.

Entities in a satellite system must interact with others remotely. These interactions may require identification prior to each requested action. If an entity can lie about its identity, or identification is not accurately validated, entities can illegitimately pose as one another in the system. If an entity in the system poses as any other entity it is said to be masquerading. The masquerading entity can violate security policies by taking unauthorized actions [6].

2.6.5 Replay.

There exists the possibility of a re-transmission of commands to the space segment of a satellite system. This replay of a specific command can occur due to a commanding protocol or by a malicious third party attempting to gain access or cause damage to a satellite. A ground station protocol resulting in the replay of a command may be the result of a previous command not properly acknowledged by the satellite, thereby prompting the re-transmission of the assumed lost command. The effects of a satellite erroneously processing duplicate commands can range from none to catastrophic loss of the satellite due to a duplicate orbit maneuver or breach of satellite security [6].

2.6.6 Software Threats.

Computer software plays a crucial role in the operation of a satellite system in both the space and ground segments. This software is susceptible to logic errors, data input handling errors, among other common programming mistakes. Additionally, operators may introduce improper configurations, resulting in security vulnerabilities or system instability [6].

2.6.7 Unauthorized Access.

Policies set forth in the operation of a space mission determine which entities should have access to specific systems and functions. Entities accessing systems or functions which violate these policies constitute unauthorized access in the system [6]. Entities may gain unauthorized access to the satellite system through a combination of other threats to the system. The abuse case presented in Section 4.4 is an example of unauthorized satellite access through a forgery attack. This attack also incorporates elements of the other threats discussed such as data replay and masquerading.

2.7 Trust

The issue of trust has become a significant concern within distributed information system architectures such as web services, cooperative computing, and mobile computing [33,34]. Trust issues are not only present in business, social and operational functions, but also in technologies used to facilitate these activities. Additionally, due to the tight coupling between a systems operational requirements and the technology used in implementation, trust relationships from the operational architecture must be modeled in the distributed information system. Specific distributed information systems must address all of the trust issues present in the operational scenario and those that arise in the technical implementation [33].

One example of modeling operational trust relationships in a distributed system implementation can be made for satellite system telecommanding. The operational function of telecommanding inherently involves a trust relationship between the satellite in orbit and a commanding ground station. The satellite in orbit must process telecommands from the ground station in a manner which preserves the functionality and security of the satellite. This trust relationship in the operational function of telecommanding must be modeled in the satellites implementation in order to satisfy the operational trust relationship.

2.7.1 Trust Management System Examples.

Research covering trust management in distributed computing has developed services and applications which accommodate trust and its related elements. Some elements which have been addressed thoroughly are reputation, and security credentials. Examples of reputation-based systems include XREP, NICE, and P-Grid. These systems aggregate the perception of entities in the system to calculate a local reputation value for a specific entity. This reputation value is then used in system policy to manage interactions with entities in the system. Credential-based systems such as X.509, PGP, PolicyMaker, and KeyNote use credentials to address the trust management problem. The primary evidence for trust in these credential-based systems is the verification of entity provided credentials. These systems enable policies which restrict access to services and resources to verified entities. [35]

While both reputation and credential-based trust management approaches address the issue of trust in distributed information systems, neither provides a general description of a trust management system, nor incorporate all of the desirable features found in current trust management research. A comprehensive description of a general trust management system is found in the works of Weiliang Zhao, Vijay Varadharajan, and George Brian [36,37,38,35]. These papers develop a general methodology for modeling trust relationships and provide a unified framework for trust management. The unified framework developed incorporates aspects of trust management from a variety of related research [35].

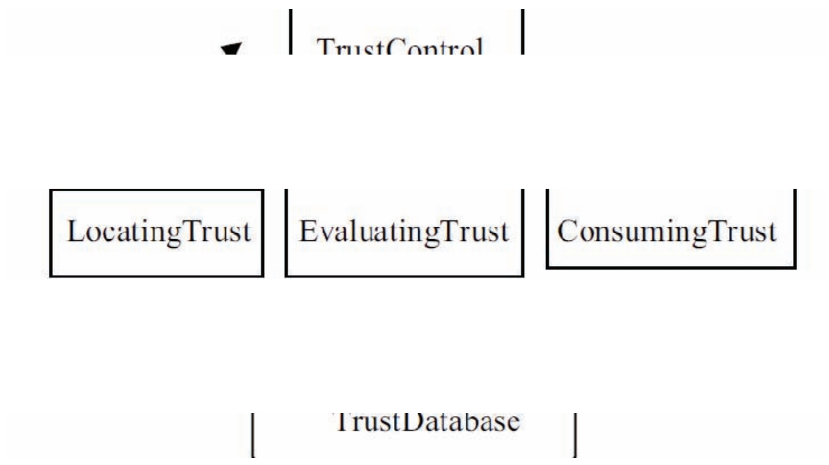


Figure 1 Trust Engine Hierarchy

The basic TMS presented by Zhao et al., referred to as TrustEngine, manages trust through the hierarchy depicted in Figure 1. This architecture incorporates all of the trust related components which may be separated from applications and handles trust requests similar to database queries. Input is made to the system as a set containing trustor, trustee, conditions, and trustee properties. The system will return a value depending on the input received. An example response is the result of a trust relationship evaluation [35].

TrustEngine consists of several components which serve trust functions or store trust data for the system. The TrustDatabase component stores trust relationship information and trust parameters. This is a persistent storage mechanism for extended storage and retrieval of trust related information. TrustControl provides overall control of TrustEngine at runtime by linking applications to the functional packages in TrustEngine. LocatingTrust performs a direct query of existing trust information from the TrustDatabase component.

The EvaluatingTrust component evaluates the current status of a trust relationship. This evaluation consists of checking whether the conditions of a trust relationship are satisfied or not. These conditions may be indicators of malicious behavior or environmental factors, and can be computed by multiple trust mechanisms. Existing mechanisms for checking trust conditions such as credential, reputation, or environment evaluation may be incorporated into the EvaluatuingTrust component.

ConsumingTrust handles the EvaluatingTrust component output. This is necessary as not all trust evaluations will be consumed immediately. Additionally, the evaluation output may require specific formatting for use by the requesting application which is handled by the ConsumingTrust component.

2.7.2 Trust Management System Design Principles.

The following is a discussion of trust management concepts as it relates to distributed information systems put in the context of satellite telecommanding. Trust has become an intrinsic part of other distributed information technology areas such as e-Business [33], and critical infrastructure protection [39]. The nature of satellite telecommanding studied here closely resembles the definition of a distributed information system utilizing the client-server model [40]. With this relationship, models for trust management and design principles in distributed information systems will be applied to telecommanding (message passing) for satellite systems.

Weiliang Zhao and Vijay Varadharajan presented a unified trust management framework which introduced general characteristics for consideration in TMS development [33]. These TMS characteristics as they relate to satellite systems are:

- Multiple Trust Mechanisms:

  Trust can be established between entities by various methods. These methods are modeled in the TMS with specific trust mechanisms. Examples of trust mechanisms are credential verification trust, reputation-based trust, and trust derived from local data. TMSs can incorporate multiple trust mechanisms in concert for a single trust decision regarding a complex trust relationship [33]. Without multiple trust mechanisms, a TMS is limited to modeling simple relationships. These simple relationships are commonly handled by an authentication protocol or other security mechanism.

- Open Nature:

  Satellite telecommand systems are open to all wireless transmissions, allowing everyone to access the satellite's physical channel remotely. As the system is open, trust relationships must be defined for known and unknown entities accessing the system. The open nature of wireless telecommand links makes trust management a crucial part of the entire system [33].

- Multiple Domains:

  Operations involving satellite telecommanding often span several networks with organizational, physical, and logical boundaries. The interconnection of these networks can be hierarchical or parallel. For example, a satellite control network is used to command a single satellite. The satellite control network consists of command terminals which are remotely located from the uplink facility transmitting signals to the satellite. The remote command terminals access the uplink facility through a terrestrial communications network. In this case, there is a hierarchical relationship between the remote command terminal and uplink facility sending signals to the satellite. There is a parallel relationship between multiple ground transmission sites. Trust relationships can be complex in the entire telecommanding system. Issues can arise in cross-boundary operations, management, and administration [33].

- Real-Time Trust:

  For most distributed information systems, real time evaluations are required for trust relationships. This is also true for satellite telecommanding systems. The dynamic trust relationships in satellite telecommanding require a real time evaluation of trust in any TMS being applied. In order to facilitate this real time trust evaluation, evidence used to calculate trust must be collected and made immediately available for a trust determination. An analysis of the relevant time frame for this trust evaluation must be conducted to ensure a "current" trust result is being used when necessary [33].

- Scalability:

  Each distributed system has a scale at which it operates. A TMS implementation must be able to scale to meet the maximum requirements of the distributed system. The scale involved with traditional satellite commanding architectures is relatively small compared to web-based distributed systems [33].

- Complexity:

  Complexity in modern distributed information systems is increased by complicated business functions and advanced technology employed in the architecture of such systems. TMSs introduced to or developed in these systems must be capable of matching and modeling the complex trust relationships involved [33].

The above items describe key areas which should be considered for trust management in distributed information systems. These characteristics are evaluated in this work for the development and implementation of a TMS for satellite telecommanding.

III. Methodology

3.1 Methodology Overview

This thesis addresses threats to the security of satellite command links through the development of a multi-mechanism Trust Management System (TMS) for satellite Flight Software (FSW). The previous chapter discusses the basic domain of satellite systems, general trust management, and computer security concepts. This chapter covers the development of a TMS incorporating interaction, policy and credential-based trust. Additionally, a satellite commanding abuse case is formulated to test the TMS. The developed TMS and formulated abuse case will then be applied to a FSW model where comparison is made between a FSW operating with and without the TMS. This chapter presents the motivation for, and explanation of key design features of the Consolidated Trust Management System (CTMS), a multi-mechanism TMS for application in satellite FSW. The definitions and assumptions associated with the CTMS, FSW model, and abuse case are also presented.

3.2 Problem Definition

3.2.1 Goals and Hypothesis.

The work presented by Zhao and Varadharajan [33] detailing a unified trust management framework provides a high-level architecture, considerations, and theory from which a TMS is developed. Additionally, the documentation related to the KeyNote TMS [41] [42] provides an excellent example of implementing a TMS, although it is limited to credential and policy evaluation. With the framework provided by Zhao and Varadharajan and the implementation lessons from the KeyNote documentation, a new TMS is developed to handle the complex trust requirements in satellite FSW.

This new TMS will be referred to as the Consolidated Trust Management System (CTMS), and incorporates multiple trust mechanisms and policies. It is hypothesized that the CTMS will enhance security in FSW telecommanding by allowing the detection of anomalous behavior from the FSW's perspective. The goal is to determine if the CTMS running within the FSW can detect the activities of a malicious ground station and respond with a pre-programmed policy and trust thresholds.

3.2.2 Approach.

To design an effective TMS for FSW telecommanding that will detect anomalous behavior, the multiple trust mechanism TMS framework proposed by Zhao and Varadharajan will be implemented with an architecture similar to that of the KeyNote TMS [33,41,42]. Additionally, the method for calculation of an Interaction Trust (I-Trust) value as proposed by Bin Yu and Munindar Singh will be adapted to provide a quantitative measure of I-Trust for entities in the system [43]. The I-Trust algorithm from Yu and Singh is extended to provide a more descriptive measure of I-Trust by basing the trust value calculation on a specific marker associated with system interactions, see Section 3.3.3.1. This I-Trust value is used as a component in the final trust evaluation of an entity, see Section 3.3.4. The I-Trust calculation and management of I-Trust values will exist as a trust mechanism within the CTMS, see Section 3.3.2.

An I-Trust marker is a class of evidence in the system which is collected by observing interactions with an entity. One example of a trust marker in FSW is the command authentication counter check result. Each command contains a value for the command counter which is compared with the satellites onboard value upon receipt. This check is the authentication counter trust marker and is modified based upon cooperation or defection interactions with the system. Additional markers can be associated with a single transaction, such as message size, or argument validation checks.

With the CTMS developed, a satellite commanding abuse case is formulated as a basis to generate policies and identify interaction markers for tracking within the system. The abuse case is a specific instance of a threat action or attack scenario applied to the satellite FSW system. An analysis of space mission threats is combined with a working FSW model to formulate specific abuse case for the system, see Section 3.4 and 4.4.

With an abuse case and CTMS complete, a test setup is established which runs FSW in an emulated environment. The test setup includes original ground station software for normal commanding operations and custom ground station software to apply the abuse case. From the test setup, comparisons are made between FSW with trust management principles applied and basic FSW with no trust management. Additionally, variations on policy options within the CTMS are compared to highlight their effectiveness and impact on the system.

3.3 Trust Management System

3.3.1 Services.

The CTMS serves as an engine for tracking and evaluating complex trust relationships in satellite FSW. The functionality of the CTMS is used as a security mechanism to support the three computer security services of confidentiality, integrity, and availability [32]. With a given system security policy, the CTMS acting as a security mechanism can be implemented to prevent, detect, or recover from attacks waged on the system [32]. This allows CMTS to monitor and support the three computer security services.

The CTMS architecture supports the confidentiality of a system by monitoring interactions secured through cryptographic means for activity which would be indicative of a key compromise. An example of such activity would be repetitive failures to successfully validate the command authentication counter. This activity could simply be logged for detection, or further compromise could be prevented by discontinuing use of the potentially compromised key. Furthermore, the CTMS architecture can be used to initiate telemetry notification of the event using a secure backup key.

System integrity is supported by monitoring interactions with ground stations and components aboard the spacecraft for potentially corrupt data. The corrupt data could be improperly formatted messages or erroneous telemetry values. The CTMS architecture could be used to detect or prevent this activity. Additionally, recovery features can be initiated through the CTMS. Logging the activity being monitored provides detection services. Further corruption can be prevented by discontinuing use of the corrupted data. Additionally, system integrity could be recovered by rebuilding corrupt data from a valid checksum.

CTMS functions can also secure satellite system availability by maintaining trust relationships between the FSW and components in the untrusted environment. If these trust relationships are degraded, the FSW can fall back to a safe state, thus preserving system availability for recovery operations. This functionality, provided by the trust management system, stems from and can be extended through the addition of trust mechanisms within the CTMS architecture.

3.3.2 Architecture.

The CTMS architecture and implementation are primarily derived from the KeyNote TMS documentation [44], while the concept of multiple trust mechanisms and additional trust management theory stems from Zhao and Varadharajan's trust management framework [33]. The development of the CTMS architecture considers the major characteristics for trust management systems from Zhao and Varadharajan [33] as they relate to the trust issues for satellite FSW.

The first characteristic included in CTMS is multiple trust mechanisms. The trust mechanisms incorporated initially into the CTMS architecture for this work are the I-Trust and credential trust mechanisms. Additionally, the CTMS architecture is flexible with the provision for additional trust mechanisms to be added as needed.

The second characteristic considered for the CTMS is to address the open nature of telecommanding in satellite systems. Illegitimate ground stations can broadcast signals or commands to an orbiting satellite. This open nature influences the development of a TMS architecture for satellite FSW and illustrates the need for TMS monitoring of open interactions.

Authentication mechanisms within the system can be used to filter known from unknown user communications, but only after some processing of the transmission is performed. Some FSW systems may not include authentication mechanisms [31]. The CTMS addresses unauthenticated communications with a trust pool for unregistered users and handles commands attributed to these entities as anonymous. Without a mechanism for authenticating users, to isolate them from the anonymous trust pool, policies cannot be implemented which discriminate between entities. This issue is addressed with the credential mechanism in the CTMS architecture.

The third major characteristic of the CTMS is the availability of real-time trust evaluations. Trust relationships in FSW are continuously changing. An example of the dynamic trust relationships in satellite systems is demonstrated with the satellite command link. Messages received through the command link are validated based upon system parameters resulting in trust evidence from the interaction. This evidence can indicate a potentially legitimate (cooperation) or malicious (defection) interaction with an entity. The resulting trust relationship must reflect an entity's cooperation or defection behavior.

The CTMS architecture provides the capability to dynamically evaluate interactions based upon markers in the system. These markers are specific characteristics of the interaction, such as a valid command authentication count. The I-Trust value for an entity is computed in real-time during interactions and is immediately available as TMS data. The policy evaluation function then uses this TMS data to compute entity trust relationships and return a policy compliance status.

The fourth trust management characteristic considered is that of scalability. Within satellite FSW, the TMS scale is limited by the resources onboard the satellite and the number of external entities involved in operations. The CTMS architecture can scale to accommodate varying types and numbers of entities paired with multiple trust mechanisms and policies. For example, all ground stations expected to communicate with the satellite can be added to the credential and I-Trust mechanisms. Trust data can then be computed individually for each ground station with proper authentication. Additionally, satellite local entities (onboard components) can each be added as members in CTMS by establishing an authentication procedure and tracking an I-Trust value for each identifiable component of interactions with them.

Lastly, the complexity of satellite systems is considered in the CTMS architecture. The complexity inherent in satellite systems through the incorporation of multiple advanced technologies influences trust relationships. As the satellite FSW integrates all spacecraft functions it serves as a mission critical subsystem and hub for processing and evaluating the trust relationships in the spacecraft. The complexities found in satellite systems trust relationships, embodied in the FSW, can be modeled with CTMS components due to its modular design.

From the KeyNote TMS implementation, all TMS functions are contained within the KeyNote Interpreter. This consolidation of trust management functions allows the TMS to be implemented with minimal complication to the overall software system. Similarly, the CTMS functions are implemented outside of the FSW application code. Trust management operations are accessed with simple function calls to the CTMS. As requests arrive to applications within the FSW, these applications will make updates to trust mechanisms as necessary to maintain trust evidence in the system. Before applications process potentially hazardous external requests, a trust determination is requested from the CTMS based upon a selected policy.

All of the considerations for design and implementation details as discussed can be seen in the general CTMS architecture Figure 2. The general CTMS architecture shows the integration of trust management modules with the satellite FSW. Additionally, interactions between the entities and components in the system are shown.

The primary function of the CTMS is to provide policy evaluations based upon trust evidence within the FSW . The components which provide this capability are the Interaction Trust Mechanism, Credential Trust Mechanism, Policy Evaluation Function, and CTMS Application Programming Interface (API).

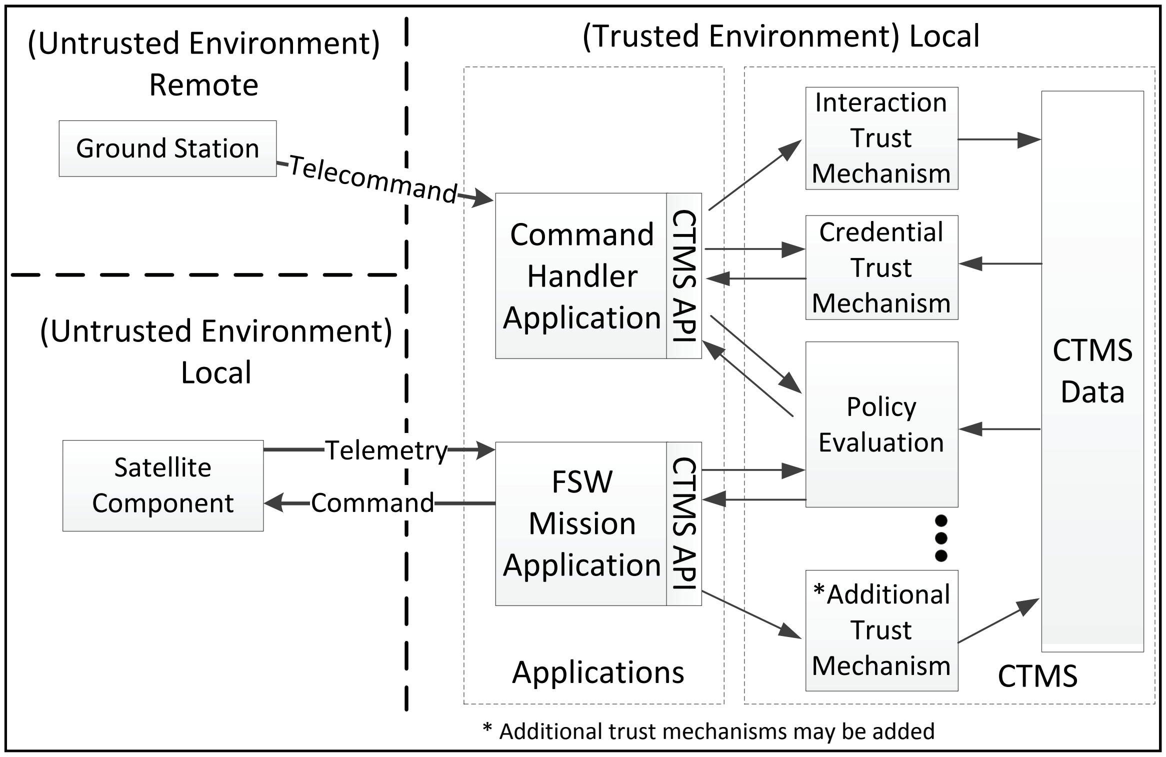

(Untrusted Environment) Remote
Ground Station
Telecommand
(Untrusted Environment) Local
Satellite Component
Telemetry
Command
(Trusted Environment) Local
Command Handler Application
CTMS API
FSW Mission Application
CTMS API
Applications
Interaction Trust Mechanism
Credential Trust Mechanism
Policy Evaluation
*Additional Trust Mechanism
CTMS Data
CTMS
* Additional trust mechanisms may be added

Figure 2 General CTMS Architecture

3.3.3 Trust Mechanisms.

The CTMS architecture is flexible and can employ many trust mechanisms. The incorporation of additional trust mechanisms to the CTMS allows more complex trust relationships to be evaluated by system policies. Much like the I-Trust mechanism monitors the behavior of entities in the system based upon interaction markers; additional trust mechanisms can be used to determine how entities should be trusted in the system. One example is an environmental trust mechanism. Environmental parameters can be evaluated against a standard in real-time with the results being evaluated by system policy. Policy defines which entities should be trusted in the system and what actions should be taken considering trust evidence in the system.

3.3.3.1 Interaction Trust Mechanism.

The I-Trust mechanism consists of functions which calculate and maintain I-Trust values for entities communicating with the FSW. Each entity being tracked by CTMS can have multiple trust markers associated with it. A separate I-Trust value is calculated for each marker associated with an entity. These I-Trust values are later used to make policy determinations in the system.

I-Trust markers are defined as key indicators in the system either inherent to the FSW or specifically added to characterize entity interactions. As previously indicated, an example I-Trust marker is the command authentication count field in a command message. If a message authentication count field does not correspond to the current value held in the satellites state, it is considered invalid and indicates the receipt of a potentially malicious command. Other examples of inherent I-Trust markers include command arguments, command time stamps, and the overall command format.

An example of an I-Trust marker added to the system specifically for the purpose of trust calculation is a consecutive command failure counter. This counter would be checked against a maximum threshold and an I-Trust value would be calculated based upon this marker. System policies can then make references to the I-Trust value computed based upon the consecutive command failure marker. An example policy using this marker is to generate an alert log entry with the date and time of the failed attempts. This log would be reviewed by satellite controllers for further investigation.

The I-Trust value calculation algorithm used in the I-Trust mechanism is largely based upon the work of Yu and Singh in the field of reputation management in electronic communities (social interaction) [43]. The I-Trust value presented here is applied to the communications in FSW and calculates an I-Trust value based upon a series of interactions. The resulting trust value is then compared with a policy limit on trust regarding the monitored marker (this limit is set for each marker in a trust policy). The result of an I-Trust value check is a trust rating for the marker, which contributes to the entity's overall trust rating. This marker trust rating is considered in an active system policy which determines how the system will react to low trust interactions.

To achieve the previously described trust-based policy enforcement, an I-Trust value is defined.

DEFINITION 1: T_{jx} is the trust value assigned by the I-Trust mechanism to entity ***j*** for interaction marker ***x***. It is required that $-1 < T_{jx} < 1$ and T_{jx} is initialized to zero.

The I-Trust mechanism calculates a trust value for entity ***j*** based upon its observation of interactions involving entity ***j*** affecting marker ***x***. Cooperation is an instance of system interaction in which the trust marker in question is positively affected; meaning the marker indicates legitimate activity. A cooperation interaction by entity ***j*** results in a net increase of the trust value with the factor **α,** while defection reduces the trust value with the factor **β**. The positive and negative associations for **α** and **β** require **α** ≥ 0 and **β** ≤ 0. The specific magnitudes of **α** and **β** are determined by the nature of the interactions taking place which modify the trust marker. Typically, trust relationships are such that trust is hard to gain and easy to lose. This results in the relationship |**α**| < |**β**|. The values of **α** and **β** can be either static or dynamic depending on the nature of the environment to which the trust system is being applied [43]. Further detail regarding the selection of suitable **α** and **β** values for the command authentication count I-Trust marker is presented in Chapter IV.

DEFINITION 2: After an interaction, the resultant trust value T'_{jx} is calculated by the algorithm presented in Table 1 which considers the previous trust value T_{jx}.

Table 1 Simple Interaction Trust Algorithm [43]

T_{jx}	**Cooperation interaction by j**	**Defection interaction by j**
< 0	$T'_{jx} = T_{jx} + \alpha(1 - T_{jx})$	$T'_{jx} = \dfrac{T_{jx} + \beta}{1 - \min(\lvert T_{jx}\rvert, \lvert\beta\rvert)}$
> 0	$T'_{jx} = \dfrac{T_{jx} + \alpha}{1 - \min(\lvert T_{jx}\rvert, \lvert\alpha\rvert)}$	$T'_{jx} = T_{jx} + \beta(1 + T_{jx})$
= 0	α	β

Table 1 presents the algorithm for computing interaction trust, which will be referred to as the simple I-Trust algorithm. Following the work of Abari and White, the simple interaction trust algorithm was tested against a confidence attack [45]. This initial testing was performed to gain an understanding of the simple interaction trust algorithm and to characterize it's performance under a specific attack scenario.

A confidence attack, or con man attack is a sequence of interactions where an entity conducts a series of consecutive cooperation interactions to elevate an associated trust rating in the system. These cooperation interactions are followed by a single defection interaction, which would result in a benefit to the con man (malicious) entity. This defection interaction also lowers the system trust value for the con man. The attackers intended result of this activity is the systems continued processing of defection interactions resulting in a net benefit to the attacker.

An initial analysis of the simple trust value calculation was performed by simulating a number of cooperation and defection interactions with entity **j**. These interaction sequences were based upon the concept of a con man attack where entity **j** interacts cooperatively Θ times before a single defection. This Simple Con man Attack (SCA(Θ)) pattern was repeated for 250 individual interactions with a graph of the calculated trust values shown in Figure 3 [45].

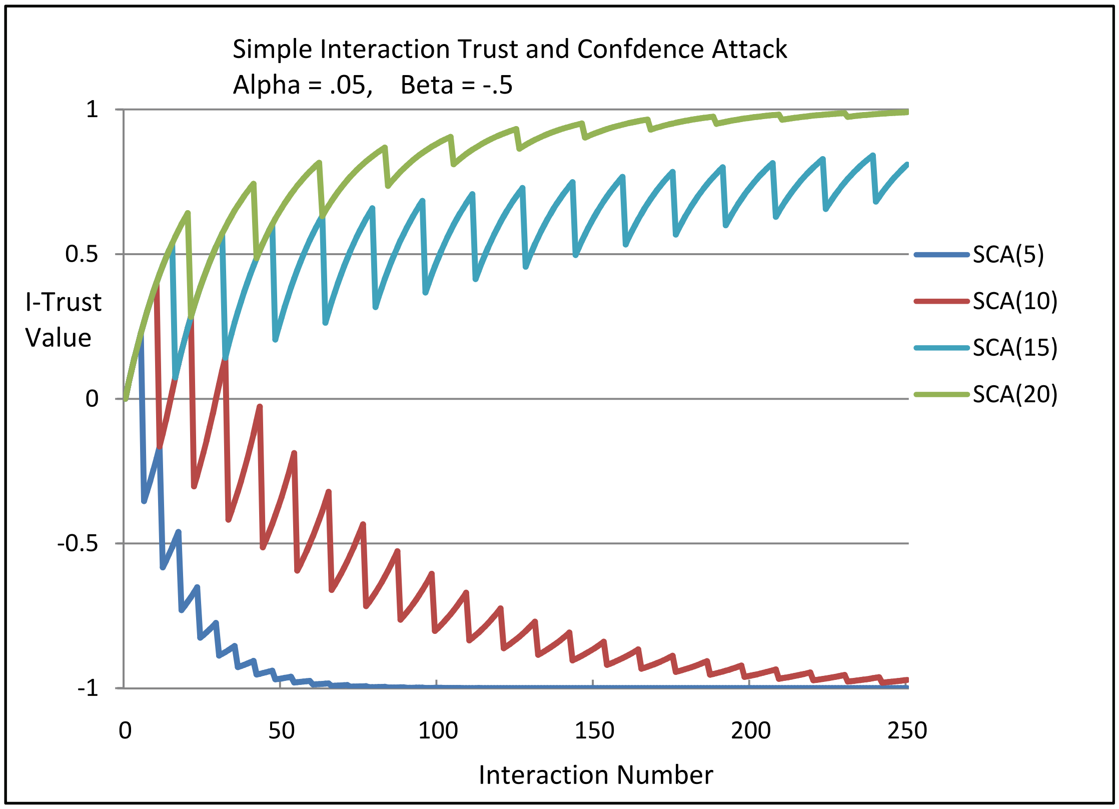


Figure 3 Simple Trust Value Graph During Confidence Attack

Figure 3 displays a graph of calculated I-Trust values throughout four different con man attack patterns. The interaction trust values for each attack pattern begin at the initialized value of zero. The interaction trust values for each pattern increase with the initial cooperation interactions and subsequently drop at the first defection interaction. For SCA(5) the interaction trust value reaches 0.23 before the first defection interaction, which results in an interaction trust value of -0.35. With SCA(20), 20 cooperation interactions are calculated before the initial defection interaction is processed. The interaction trust value for SCA(20) before the initial defection interaction is 0.64 and is 0.28 after. Figure 3 shows the simple interaction trust value can converge to a high value with extended intervals between defection interactions [45].

This characteristic of the simple interaction trust algorithm to maintain a steady positive trust value with a known number of defection interactions is important to modeling certain trust relationships. In particular, this method is used to model trust relationships between a satellite and ground stations based upon the command authentication count marker. This is due to the nature of satellite telecommanding where legitimate ground stations may have a number of defection interactions during a telecommanding encounter.

To make the system of trust value calculation resistant to potential confidence attacks, values of **α** and **β** may be dynamically adjusted based upon entity interaction and the current trust value. This method follows work presented by Abari and White [45].

The initial value for **α** is preserved as **α_0** during the entire series of interactions by the I-Trust mechanism. The algorithm for **α** and **β** determination to achieve a con-resistant trust value calculation is shown in Table 2. The algorithm described in Definition 3 along with the trust calculation in Table 1 and Table 2 will be referred to as the extended I-Trust algorithm.

DEFINITION 3: **α** and **β** are determined for con-resistant trust value calculation by the algorithm in Table 2, where **C** is a constant $0 < C \leq 1$:

Table 2 Extended Interaction Trust Algorithm

Cooperation interaction by j	**Defection interaction by j**
$\alpha = \min(\alpha + \gamma_c(\alpha_0 - \alpha), \alpha_0)$	$\alpha = \alpha(1 - \lvert\beta\rvert)$ $\beta = \beta - \gamma_d(1 + \beta)$
$\gamma_c = 1 - \lvert\beta\rvert$	$\gamma_d = C \times \lvert T_{jx}\rvert$

Figure 4 is a plot containing the basic test results of the extended I-Trust algorithm during a confidence attack. The interaction patterns used in this initial evaluation of the extended I-Trust algorithm are the same as those used for the simple I-Trust algorithm shown in Figure 3. Figure 4 shows the I-Trust values are more severely impacted by defection activity and none of the interaction patterns converge to a high trust value. The extended interaction trust algorithm may be suitable for interactions which provide benefit to malicious entities for repeated abuse.

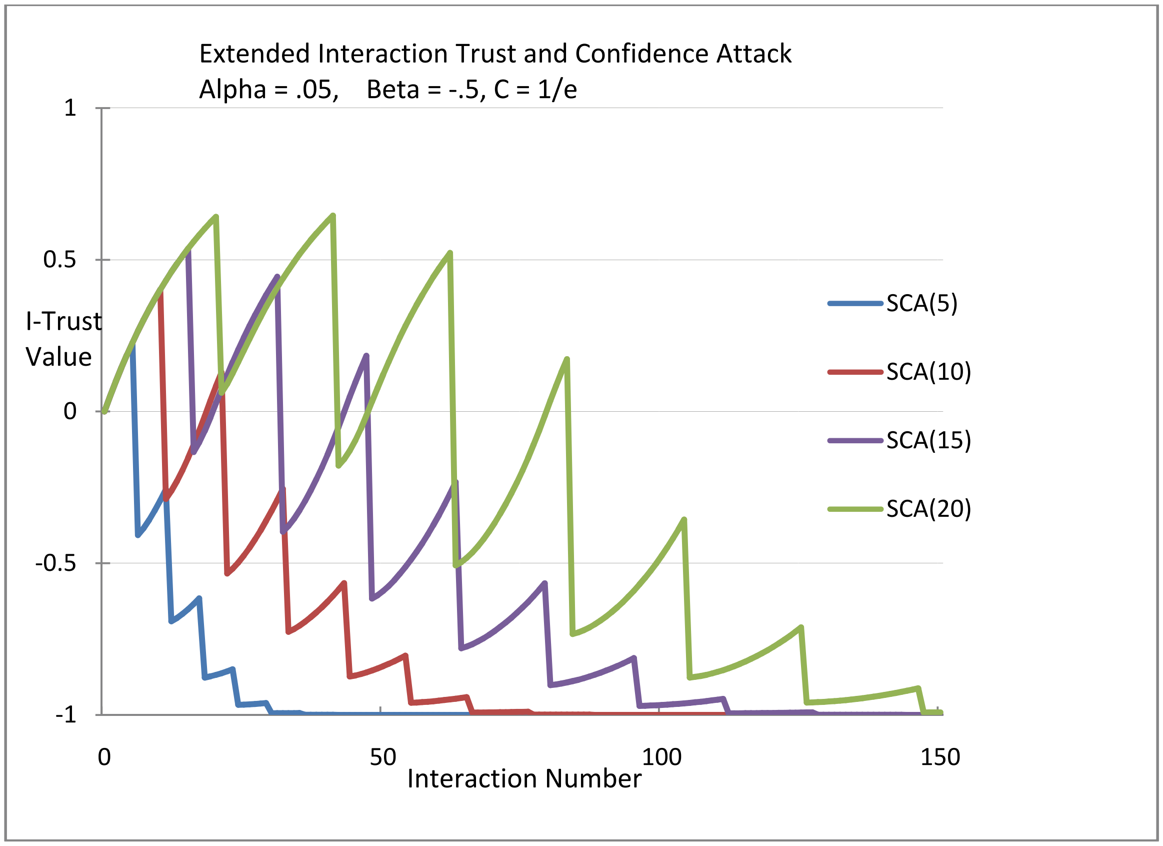


Figure 4 Extended Trust Value Graph During Confidence Attack

For the I-Trust mechanism, the I-Trust value corresponding to each marker tracked for entities in the system is initialized to zero. The I-Trust value alone may not provide enough information determine how an entity will be trusted by the system. This value is simply an indicator of one entity's behavior based upon evidence concerning a specific trust marker. How the I-Trust value will be interpreted is up to the system policy utilizing the value, see Section 3.3.4.

Applications within the FSW call upon to the I-Trust Mechanism to update trust evidence based on system interactions. Once an interaction has been sent to the I-Trust mechanism, system policy can be evaluated based upon the current system status. FSW applications use the result of the system policy evaluation to process interactions.

The system policy making use of I-Trust values defines a threshold for acceptance. The threat being addressed by any policy must be characterized and resulting parameters required for I-Trust calculation must be established by the developer. To properly identify the threat, the I-Trust calculation must fit within the threat model and accepted use of the system. Chapter IV and Appendix A provides additional discussion relating to system characterization and policy determination to mitigate threats to satellite systems with the CTMS architecture.

3.3.3.2 Credential Trust Mechanism.

The credential trust mechanism processes authentication credentials to provide cryptographic authentication within the FSW. The use of cryptographic authentication identifies entities with a high degree of assurance. The credential trust mechanism can be implemented with any one or a number of different authentication protocols. The

CCSDS has recommended the HMAC authentication algorithm SHA-1 as a standard for message authentication in satellite systems [30].

Some satellite FSW does not have authentication services included in system code [31]. Authentication services can be integrated into FSW with a robust TMS architecture or dedicated authentication service. A dedicated authentication service would require developers to implement an authentication protocol and integrate its functionality into the FSW. The credential trust mechanism in the CTMS architecture provides entity authentication for the FSW through standard function calls. This mechanism will receive a credential provided by an entity and determines if the credential is valid thereby identifying the entity. With user (entity) authentication available to the FSW, policies can be implemented which consider the authentication status of entities interacting with the system. This authentication is critical to the performance of security related protection functions with CTMS architecture.

3.3.4 Policy Evaluation.

The CTMS serves as a security mechanism by evaluating the systems security policy. These evaluations are performed by the policy evaluation function in the CTMS architecture. This function requires the system security policy to be stated in quantifiable terms which can then be evaluated for compliance. Additionally, the policy evaluation function must have access to the objects and variables referenced in the system security policy.

The policies evaluated can model complex trust relationships making full use of all trust mechanisms and trust evidence collected in the system. The result returned from the policy evaluation function is used in the FSW application as the final trust rating for the entity in question. The policy evaluation function gives FSW applications a mechanism for enforcing system security policies.

3.3.5 Consolidated Trust Management System (CTMS) API.

In order for FSW applications to make use of the trust management system, the CTMS Application Programming Interface (API) must be incorporated into application code. The CTMS API consists of function specifications which are used to update trust evidence information in trust mechanisms, gather information from trust mechanisms, and request policy evaluations. The API also specifies where CTMS functions should be placed in application code to perform the desired trust management activity. With the applications making use of the CTMS API, trust evidence and policy evaluations may be processed in the FSW.

3.4 Abuse Case Development

The specification of an abuse case describes a complete and detailed set of interactions that result in actual harm to the system. A maximal abuse case has been characterized as gaining complete control of the system through an abuse of privileges. The maximal abuse is not always necessary to characterize an abuse of the system. Simple abuses minimally compromise the privilege necessary to accomplish an intended harm on the system [46].

Within an abuse case, actors are described by their characteristics. The critical characteristics necessary for modeling an actor in an abuse case include the actors resources, skills, and objectives [46]. For this work, actors in the satellite telecommanding abuse case must be modeled with characteristics relevant to the satellite systems domain.

The purpose of an abuse case is to describe a family of undesirable interactions with the goal of reducing security requirement oversights and design flaws. The abuse case definition includes many abstract transactions which may be used to accomplish a single abuse of the system [46]. Specific features or components in a system which may be exploited are selected and included in the abuse cases description.

The concept for a satellite commanding abuse case is derived from the CCSDS report concerning threats to space systems and known computer system intrusion techniques [6]. Additionally, related research on telecommanding security refers to a specific attack case for satellite commanding communications.

The related attack case involves a malicious entity attempting unauthorized access to a satellite command link through a forgery attack. This forgery attack requires an attacker to possess knowledge about the spacecrafts orbital and physical channel access parameters. Additionally, the attacker is modeled to have access to the command message structure, and can transmit modified messages to the satellite [47].

The steps of the referenced forgery attack begin with the attacker intercepting legitimate satellite command messages. These messages are then analyzed for the underlying protocols. Once identified, fields in the message relating to satellite access are modified in a manner which will allow the attacker to successfully command the satellite [47].

The objective of the attacker in the previous scenario is to gain control of the satellite. It is this activity that this thesis serves to address with the CTMS architecture. Further discussion regarding the specific forgery attack satellite telecommanding abuse case used in this work is presented in Chapter IV.

3.5 System Policy Development

The system policy used in this research will be based upon the abuse case activities being tested in the system. Through this relationship, system policy plays a critical part in addressing security in the satellite system. One of the primary benefits of CTMS is to provide mechanisms with which to implement system policy. For example, the CTMS interaction trust mechanism will allow system policies to be implemented which take into consideration a users interactions. Additionally, the system policies used for testing are designed to demonstrate the logging, detection, and prevention of malicious activities utilizing the CTMS.

3.6 Satellite Test Environment

A functional satellite system is required to support implementation of the CTMS architecture to satisfy the goal of this thesis. The final implementation and associated testing demonstrates the feasibility of the CTMS architecture. This testing is also performed to demonstrate the use of a CTMS implementation to support satellite system security. The necessary components for constructing this system are, realistic satellite FSW, realistic FSW executing environment, and a functional ground station to communicate with the emulated satellite.

An example of such a satellite system test environment known as the *Flight-Cyber Vulnerability Assessment Testbed* has been developed and implemented by the Aerospace

Corporation (funded by the USAF and Aerospace Internal Research and Development) [48]. The vulnerability assessment testbed was developed to perform specific experiments supporting vulnerability assessments for USAF programs of record. The vulnerability assessment testbed consists of two major components, the Unit Under Test (UUT), and testbed supporting infrastructure. The UUT is the satellite being tested, and is the subject for experimentation. The testbed supporting infrastructure provides all necessary hardware and software to communicate or otherwise interface with the UUT [48].

The satellite test environment developed for this research includes the major components found in the Aerospace testbed. This research testbed is required to support CTMS testing once integrated into a satellite system. The testbed components used in this work were selected to provide a realistic environment for satellite FSW security testing. My test environment implementation details, including FSW development, ground station setup, and test components, are presented in Chapter IV.

3.7 Summary

This chapter presents the design features of the CTMS for application to satellite FSW. CTMS represents a unification of trust management theory and exiting TMS implementation architecture for the purpose of enhancing satellite system security. The approach of CTMS is to merge concepts from a generic TMS framework into satellite FSW to mitigate threats associated with a specific abuse case.

The CTMS architecture incorporates multiple trust mechanisms and a policy evaluation function to perform trust evaluations of entities within FSW. The I-Trust mechanisms fundamental objective is calculating trust values for entities based upon specific interaction markers. The credential trust mechanism provides authentication by evaluating credentials provided by entities in the system. The component trust values and information within the FSW are used by the policy evaluation function to define the systems response to activity in the system. The CTMS API is used to access this functionality from satellite FSW applications.

The concept of abuse case modeling is applied to the satellite FSW to test the security of critical components and to evaluate the CTMS. System policy follows the abuse case to detect and prevent malicious activities. Additionally, the specification of a satellite test environment is presented to serve as a realistic proof of concept for the application of trust management practices to secure satellite systems.

In summation, the CTMS architecture once implemented is expected to improve security in satellite systems by detecting and preventing specific abuse cases. In order to evaluate the feasibility and performance of the CTMS architecture, Chapter IV presents a CTMS implementation in realistic satellite FSW. Finally, Chapter IV describes testing of the CTMS implementation with an abuse case and system policy.

IV. Analysis and Results

4.1 Chapter Overview

This chapter documents the study of how trust management concepts can be used to protect a satellite system from a specific abuse case. Satellite system protection is shown as the outcome of abuse case testing on satellite Flight Software (FSW). The specific trust mechanisms within the Consolidated Trust Management System (CTMS) architecture are characterized to determine their suitability for implementation in actual systems. Additionally, implementation specific results of FSW performance are presented to demonstrate the feasibility of trust management integration into satellite systems.

The goal of the current experimentation is to determine the applicability of trust management concepts in general, and CTMS specifically, to address satellite system threats embodied in the forgery attack abuse case. The primary hypothesis is that the CTMS will improve security in FSW telecommanding. This improvement in security is measured through the detection and prevention of a forgery attack with the CTMS.

This research does not focus on completely securing the satellite system emulated in the test environment; rather, it develops a methodology which can address specific concerns regarding satellite security. This methodology includes a specific architecture with which to incorporate trust management concepts into satellite system FSW. The experiments designed for this research apply the abuse case, system policy, and CTMS to the existing FSW as a proof of concept.

4.2 Test Environment Setup

A functional satellite system is required to perform the work set forth in this thesis. To serve as a functional model of a satellite system, realistic FSW, dedicated flight hardware, and satellite commanding ground station software have been acquired and configured to emulate a satellite system. An overall view of the satellite system emulated to conduct this work is shown in Figure 5.

The satellite system consists of a single satellite, and two command ground stations. One of the command ground stations serves as a legitimate ground station which is authorized to command the satellite. The second is a malicious ground station which will perform attack behavior as described in the abuse case.

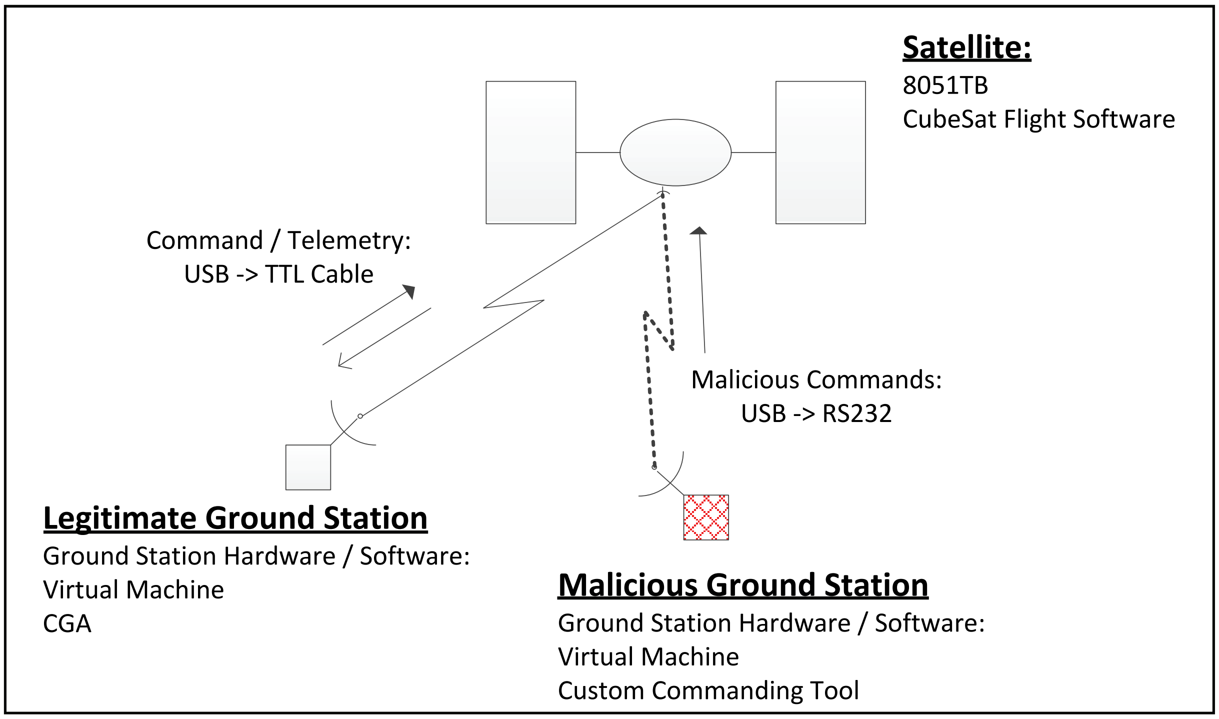


Figure 5 Test Satellite System

The satellite hardware is emulated with a microcontroller Test Board (TB) which is logically identical to the flight processor and memory for some CubeSat missions. The microcontroller test board contains a processor which runs the 8051 instruction set. Along with the 8051 CPU, memory and digital communication peripherals are located on the TB. The primary communication peripheral used is a Universal Asynchronous Receiver Transmitter (UART) which is connected to a Recommended Standard 232 (RS232) transceiver onboard [49].

The ground station hardware consists of a single computer workstation. The computer workstation divides and shares hardware resources with multiple virtual computer workstations (virtual machines). These virtual machines run the command ground station software which communicates with the emulated satellite/TB.
The ground stations utilize USB ports and signal converters to connect with the RS232 port on the TB.

The legitimate command ground station in the satellite system is modeled by a virtual machine running Common Ground Architecture (CGA). Mission unique components present in CGA were specifically designed to interact with the FSW. The CGA ground station receives telemetry from the satellite and displays system status in formatted tables. Additionally, commands can be sent from the CGA ground station to demonstrate functionality of the satellite FSW and hardware in the emulated environment.

The malicious ground station in the satellite system is implemented by custom commanding software. This commanding software was written to communicate with the satellite and perform malicious actions as described by the satellite commanding abuse case. The custom commanding software also implements new commands added to the FSW in conjunction with this thesis research.

The satellite FSW is modeled after software currently operating on CubeSat missions. The FSW is written in the C programming language with specific 8051 assembly code for system initialization and critical operations. The FSW was developed in the Keil Integrated Development Environment (IDE); which manages source code and compiles, links, and flashes complete software to the TB.

To facilitate debugging of the system and to view transmissions between ground stations and satellite, a logic analyzer was connected to monitor UART signals on the test board. The logic analyzer used for this testing was a USBee ZX module which reports the transmit and receive signals on the communication path between the TB and ground stations. The entire test environment setup described is shown in Figure 6.

Additional debugging and data output from experimentation was provided by a diagnostic port on the TB. This port served as a window into the operation of the FSW and reported real time system status directly from the TB. CTMS and FSW status was observed from this port along with any system errors generated during testing. An illustrated test using this setup is presented in Appendix B.

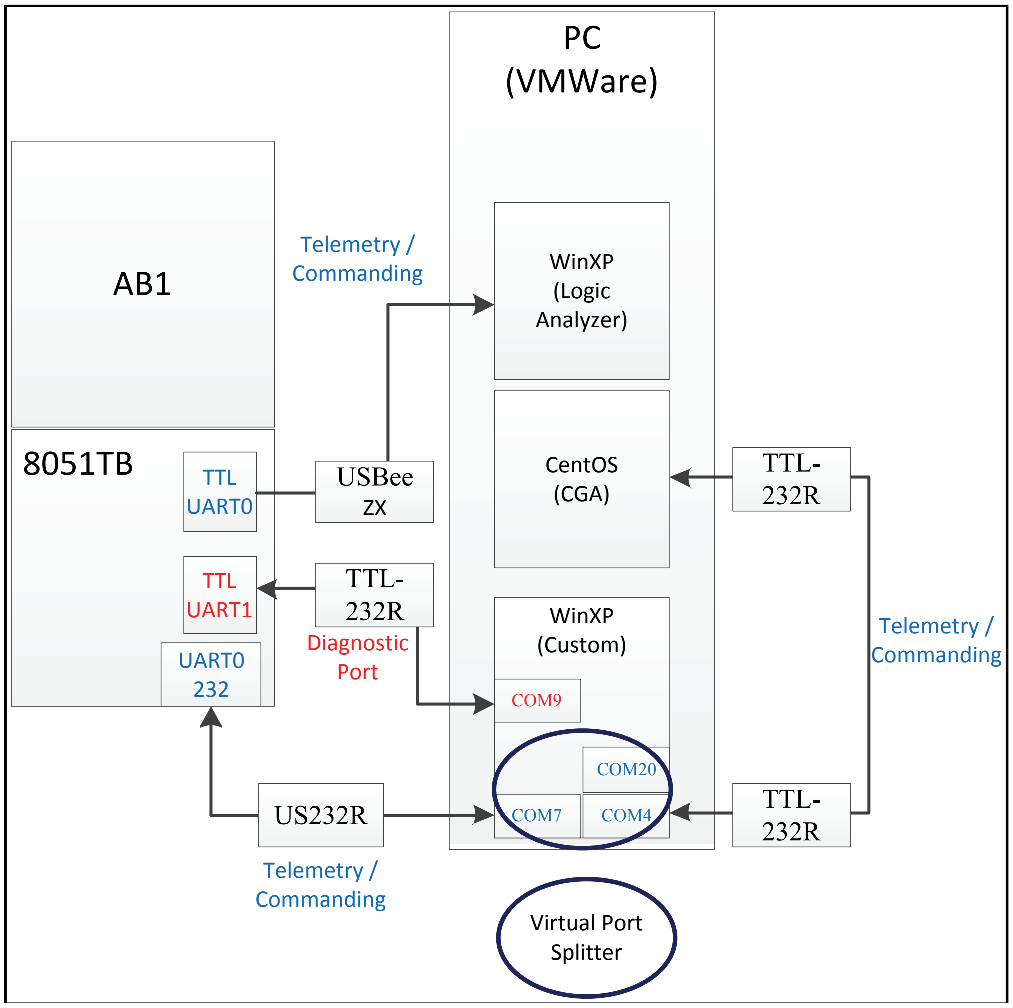


Figure 6 Test Environment Setup

4.3 Trust Management System Integration

The CTMS architecture discussed in Section 3.3.2 is integrated into the CubeSat FSW directly. The CTMS consists of functions written in the C programming language, which are compiled into the FSW. These functions are referenced by FSW applications with the CTMS API to handle trust management tasks. The modified FSW containing CTMS Application Programming Interface (API) calls, CTMS functions, and system policy are then compiled to a single binary and flashed onto the 8051 test board.

These actions would be accomplished during the development and testing phase for actual satellite mission. The complete satellite and FSW incorporating all safety and security features would be fully integrated and tested prior to launch. The following subsections describe the setup and operation of the two trust mechanisms, CTMS data storage, policy evaluation function, and CTMS API.

4.3.1 Interaction-Based Trust Calculation.

Calculations for interaction-based trust used in the CTMS implementation are performed as described in Section 3.3.3.1, with static values for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. The I-Trust values are calculated for every command interaction, however, cannot be attributed to specific actors due to lack of user authentication in the basic FSW. The lack of authentication leads to I-Trust values which characterize the trust of anonymous entities in the system. The resultant I-Trust values are used in conjunction with system policy to make a final trust determination and define the systems response commands being received from anonymous entities.

The command access security policies defines threshold values for I-Trust and states how these values will be acted upon in the system, see Section 4.5. The first step in setting up this policy is to configure the I-Trust mechanism. The initial I-Trust parameters are set to ($T = 0, \boldsymbol{\alpha} = 0.05$, $\boldsymbol{\beta} = -0.2$) for the command authentication count marker based upon an initial characterization of the I-Trust algorithm. Further discussion regarding the optimization of simple I-Trust parameters is found in Appendix A. With these parameters set, the I-Trust value for this marker will fall below -0.5 after four consecutive defective interactions. Section 4.7.2 discusses in detail the rationale behind why these parameters were chosen and their effect on the system.

The only I-Trust marker utilized during experimentation was the command authentication count marker. Additional markers were considered, however were not utilized for experimentation. Table 3 shows a list of markers which may be used in the CTMS system as implemented in this work.

Table 3 CTMS I-Trust Marker List

Marker	**Evidence**	**Attributable**
Authentication Counter	Identify poor connection or attempt on authentication counter	*NO**
Check Vector	Identify attempt on authentication crypto key	*NO*
Current Password	Identify attempt on password, indicates compromised authentication crypto key	*YES***

* If the command authentication counter is checked after credential trust authentication of a message the command authentication attempt can be attributed.

** If the credential trust mechanism identifies an invalid password after successfully comparing the check vector then the attempt is attributed to the crypto key used to encrypt the credential.

Both the simple and extended algorithms for calculating interaction trust were implemented in the course of this work. Only the simple interaction trust algorithm was suitable for computing interaction trust using the authentication count marker. This is due to the nature of the radio link used for satellite commanding, which is discussed further in Section 4.7.1.

The simple interaction trust algorithm is suited for characterization of interactions with no incentive for repeated abuse after positive interactions. This is the case for satellite commanding and the command authentication counter. The extended trust algorithm may work well for interactions which benefit the attacker for presenting defective interactions after consecutive cooperation interactions. This would be suitable for the protection of features prone to repeated abuse, i.e., the attacker gets benefit from failing the marker check. One example in the context of satellite systems would be for an attacker, once gaining access, to probe a satellite with known good commands for a number of interactions and then send a malformed command in an effort to disrupt the satellite.

4.3.2 Credential Trust Evaluation.

The credential trust mechanism implemented for experimentation utilized the Advanced Encryption Standard (AES) cryptography algorithm published in the Federal Information Processing Standards Publication 197 [50]. The AES source code used in this work was adapted from an open source implementation by Niyaz PK [51]. With the AES algorithm implemented as part of the CTMS, cryptographic verification of an authentication credential was possible in the FSW.

AES was selected to provide cryptographic authentication due to its availability and the possibility for re-use in the system. The AES functions implemented for authentication can also be used to add encryption to the commanding or telemetry communications to and from the satellite. Additionally, AES is the CCSDS proposed standard for encryption in satellite systems [30]. With a cryptographic algorithm implemented, an authentication credential was developed for processing by the credential trust mechanism.

The authentication credential introduced for testing purposes consisted of the following elements: key index, check vector, and an encrypted password component. The key index was a value used to select the AES key for decryption of the password component of the credential. The check vector was a random value used to introduce a random component to the message and to serve as a check upon decryption of the password component. The encrypted password component of the authentication credential consisted of a check vector, current authentication nonce, and next authentication nonce. Figure 7 shows the structure of the authentication credential.

The check vector in the encrypted password component was set to the same value as the vector in the authentication credential. Upon decryption, the check vector from the encrypted password component was compared with the check vector from the authentication credential. If the two vectors matched, the message was properly decoded with the key identified by the key index. A positive check of the initialization vector is the result of a valid entity passing the credential or an invalid entity replaying the credential.

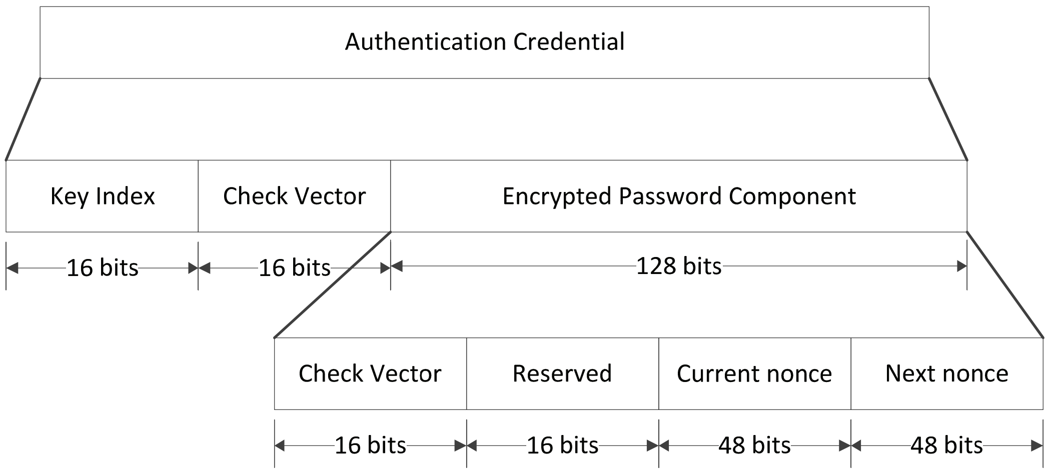


Figure 7 Authentication Credential Structure

The current nonce field in a credential was compared with the identification nonce stored in the credential trust mechanism. If the current nonce from the encrypted password component matches the internal identification nonce, then the credential is deemed to be valid. Each key index had an identification nonce associated with it for entity identification. A valid check of the credential was then associated with the key index and any activity related to this credential was attributed to the associated key index.

The next nonce field contained the value to be stored as the identification nonce in the credential trust mechanism for the given key index. The identification nonce was replaced with the next nonce value after processing the current credential. The next nonce must be different from the current nonce. If the current and next nonce are the same value in this system, the authentication credential could be successfully validated during a replay.

With the credential trust mechanism, CTMS can validate entities in the system and associate them with trust information. Entities authenticated by the credential trust mechanism were associated with CTMS members as defined in Table 4. Additionally, secure functionality can be built into the system, which can be used to mitigate system threats. An example of this is demonstrated by the experiments presented in Section 4.6.

Table 4 Credential Trust List

CTMS Member	**AES Key Index**	**Authentication Password Index**
0 - Anonymous	N/A	N/A
1 - Administrator	0	0

The authentication credential structure and associated validation mechanism was developed for use in this research as a proof of concept for a generic authentication mechanism. As such, extensive characterization and cryptanalysis of this implementation was not performed. A validated cryptographically secure algorithm and implementation should be used for credential generation and authentication in the credential trust mechanism for a flight ready system.

4.3.3 Trust Data Storage.

The Trust Data Storage component of the CTMS was used to maintain the systems current status. Each entity interacting with the system, whether authenticated or not, was categorized into CTMS members. The CTMS implementation contained data regarding CTMS members and their nominal mapping to ground station ID; see Table 5. Additionally, I-Trust data associated with each marker and entity was also stored in the Trust Data Storage component; see Table 6.

Table 5 CTMS Member List

CTMS Member	Ground Station ID	Role
0	0001	Anonymous commanding
1	0002	Administrator Level Authenticated Commanding

Table 6 I-Trust Member Evidence List

CTMS Member	Marker ID	I-Trust Marker
0 - Anonymous	0	Authentication Count I-Trust Marker Simple
1 - Administrator	0	Authentication Count I-Trust Marker Simple
	1	Credential Check Vector I-Trust Marker Simple
	2	Credential Current Password I-Trust Marker Simple

4.3.4 API.

The CTMS API implemented for testing consisted of function prototypes for the credential trust mechanism, I-Trust mechanism, and policy evaluation function. The credential trust mechanism returned the credential validation status for a given ground station ID and credential. The I-Trust mechanism did not return data directly to the calling function, however, it updated I-Trust data for a given ground station ID, marker ID and interaction result (cooperation or defection). The policy evaluation function returned the result of a policy evaluation for a given ground station ID and policy identifier.

Specific API calls were made in the command handler application, as it was being tested by the commanding abuse case. The CTMS API calls implemented in the FSW command handler application were to the I-Trust mechanism, credential trust mechanism and policy evaluation function. The I-Trust mechanism was called to update trust evidence for an entity in the system using the command authentication count marker. The credential trust mechanism was used to implement the secure unlock command discussed in Section 4.6. The policy evaluation function was used to prevent the abuse case presented in Section 4.4.

4.4 Implementation Abuse Case

Normal use for a satellite system requires commands to be transmitted from a ground station to the satellite. These commands must be processed aboard the satellite to maintain the system and to perform the primary mission. The design of command handling systems for satellites which only incorporate this simple use case with little regard for misuse or abuse of the system may lead to vulnerabilities in the system.

The scenario presented here is the result of applying the abuse case development methodology presented in Section 3.4 to the system being used to implement and test the CTMS architecture. This abuse case is modeled for the satellite telecommanding subsystem. The specific component in the telecommanding subsystem that may be exploited is the command handler application of the satellite FSW.

For this case, a third party either has access to command formatting information, or can intercept transmissions emanating from the primary satellite ground station. Once a command session has been recorded, satellite specific information is recovered from the command link data. This information is then replayed with modified/malicious values in an effort to gain access to the satellite.

The nature of the basic FSW is such that the simple replay of a previously transmitted authenticated command fails. This is due to the incremental nature of the command sequence, where previously executed commands will not be processed. These design characteristics of the FSW, however are vulnerable to a modified replay attack.

The specific abuse case is a forgery attack by a malicious ground station. This forgery attack is the replay of a previously transmitted legitimate message by an attacker. In an effort to guess the dynamic command authentication counter onboard the satellite, the authentication count field for the illegitimate message is incremented during the replay in an effort to brute force the authentication count in the FSW. The intent is to have a malicious command processed. See Table 7 for a summary of the steps involved in the abuse case.

Table 7 Abuse case steps

Step	Action
1	Record commanding session or otherwise acquire command header and format information
2	Transmit desired malicious command in an attempt to have it processed by the satellite.
3	If the command execution fails at the satellite, increment the authentication count in the command message and resend.
4	Continue to increment the authentication count in the command sequence until the command is accepted.

The abuse case activity described in Table 7 is similar to the model of attack behavior against a space communication link published by researchers investigating command link attack detection [47]. This pattern is also similar to that taken by attackers to exploit terrestrial software and networks [52].

4.5 Implementation System Policy

The system policies defined for this work were formulated to address the satellite telecommanding forgery attack. The broad system security policy requires only legitimate commands be processed by the satellite. This security policy must be enforced by a security mechanism. The CTMS implementation is the security mechanism which was used to address this broad system security policy in the following experiments.

The detailed system security policies used for CTMS implementation testing are shown in Table 8. The first system security policy requires an alert to be logged once the I-Trust value for the authentication count marker reaches -0.5. The result of this policy is attack activity detection based upon interaction trust calculation. As each command is received, the I-Trust value is updated based upon a check of the command authentication count. Failed checks will reduce the I-Trust value, while successful checks increase the value.

Policy 2 is aimed at preventing the same attack activity detected with Policy 1. Once the I-Trust value for the general ground station based upon the authentication count marker reaches -0.5 all commands from anonymous ground stations will be rejected. This policy results in denying unauthenticated users access to the system. As this policy does not specify the authentication of ground stations, it results in denial of service for all ground stations. This issue is addressed with Policy 3.

Table 8 Implemented Policy Options

Policy	**Implications**
1. Trust Management Event Logging Only	**Credential Trust not required**
Description: Command authentication count is checked upon receipt and I-Trust value is calculated for authentication count marker. Once the I-Trust value for authentication count reaches -0.5 an alert is logged indicating excessive invalid command attempts.	P- fewer satellite resources required P-legitimate ground station is alerted to review detailed logs N-malicious ground station may tamper with logs if access is acquired
2. Trust Management Event Logging and Prevention	**Credential Trust not required**
Description: Command authentication count is checked upon receipt and I-Trust value is calculated for authentication count marker. Once the I-Trust value for authentication count reaches -0.5 command processing is halted for anonymous users and an alert is logged indicating excessive invalid command attempts.	P-malicious behavior is prevented N-denial of service without entity authentication
3. Trust Management Event Logging, Prevention and Recovery	**Credential Trust required**
Description: Command authentication count is checked upon receipt and I-Trust value is calculated for authentication count marker. Once the I-Trust value for authentication count reaches -0.5 command processing is halted for anonymous users and an alert is logged indicating excessive invalid command attempts. The legitimate ground station must unlock satellite commanding and the CTMS via credential trust mechanism to resume commanding operations.	P-legitimate ground station is alerted to review detailed logs P-malicious behavior is prevented P-additional features extend policy and security options N- additional satellite resources are required N-malicious ground station may tamper with logs if access is acquired

P - Positive attribute for system policy
N -Negative attribute for system policy

Policy 3 extends the second policy with a provision for resuming satellite commanding with credential authentication of entities. Again, once the authentication count I-Trust value reaches -0.5, all commands from anonymous ground stations will be rejected until the system state is acknowledged. The system state is acknowledged and the trust management system is reset by an authenticated user with a secure CTMS unlock command. This policy and implementation is intended to demonstrate that FSW with CTMS can detect and respond to a low trust commanding interaction pattern. An example test of Policy 3 is presented in Appendix B.

The system response based on this policy is defined in the command processing code (command handler) of the FSW. The command handler filters commands from un-trusted IDs and logs un-trusted commanding interactions with the CTMS functions. Logs pertaining to CTMS status are stored and can be relayed via telemetry to ground controllers.

4.6 Experiment Design

In order to evaluate the performance of the trust management system and implement command access policies, three FSW builds were developed and tested. The first FSW build, referred to as the (Basic FSW), is based upon a CubeSat FSW implementation without trust management. The second FSW build (FSW-A) expands on the Basic FSW with an initial CTMS implementation and the first two system security policies. The final FSW build (FSW-B) implements the CTMS architecture including the credential trust mechanism and the third system security policy.

With the credential trust mechanism, FSW-B implements the third security policy which requires entity identification. This identification is performed through the addition of a secure unlock command to the FSW command handler application. The secure unlock command utilizes the authentication credential presented earlier to identify a legitimate ground station and restore access to anonymous commanding of the satellite. Table 9 illustrates the three FSW builds with their features, policies, and procedures used for testing.

Table 9 Experiment Design

	Build Features	**Test Policy**	**Test Procedure**
Basic FSW	Basic FSW + No modifications	**Broad System Policy:** Only legitimate ground stations should command the satellite	Apply abuse case and record results
FSW-A	Basic FSW + CTMS without Credential Trust	Policy 1 and 2 implemented with CTMS; see Table 8	Apply abuse case and record results for each policy
FSW-B	Basic FSW + CTMS with Credential Trust and secure unlock command	Policy 3 implemented with CTMS; see Table 8	Apply abuse case and record results; execute secure CTMS reset command and record results

The abuse case was presented to each FSW with results shown in Section 4.7. The outcome of the tests are presented with a focus on determining if an outsider can successfully intrude on a commanding session or otherwise access the satellite using the abuse case. An illustrated example of the experiment procedure for FSW-B and Policy 3 is presented in Appendix B.

4.7 Experiment Results

This section presents experimentation results from the abuse case scenarios, comparing the three FSW builds. Additionally, characteristics of the FSW builds are presented to support conclusions regarding the feasibility of implementing the CTMS architecture in satellite systems. Lastly, performance of the implemented CTMS architecture is presented as it relates to the accurate detection of the forgery attack on the satellite system.

The Basic FSW was exposed to the commanding abuse case in the first experiment. This experiment resulted in a system compromise by the malicious ground station executing commands on the satellite. This activity is prohibited by the general system policy which limits access to the legitimate ground station.

FSW-A with Policy 1 was utilized in the second experiment where the satellite was challenged by the commanding abuse case. The system was again compromised, however the CTMS implementation successfully reported the malicious activity. The broad system policy to deny unauthorized access to the satellite was violated, however test Policy 1 was successfully enforced.

For experiment three, FSW-A with Policy 2 was tested with the forgery attack. The malicious ground station was unable to execute commands on the satellite. However, by blocking the malicious ground station Policy 2 also caused a denial of service for all ground stations. This denial of service is due to the lack of entity authentication in the FSW-A build. The broad system policy to deny access to malicious ground stations was enforced along with Policy 2.

The fourth and final experiment utilized the FSW-B build, which implemented system security Policy 3. When the forgery attack was applied to this FSW build, the malicious activity was detected through the I-Trust mechanism with the command authentication count marker. Upon detection of the activity, an alert was logged and anonymous commanding was disabled. These actions satisfied the general security policy. A secure CTMS unlock command was then transmitted from a legitimate ground station with identifying credential, which was subsequently processed by the satellite. Normal commanding operations were restored on the satellite with the secure CTMS unlock command. The fourth experiment demonstrated the FSW-B successfully enforced Policy 3, which prevented malicious commanding of the satellite. The malicious commanding activity presented in these experiments is a specific forgery attack presented in the telecommanding abuse case, Section 4.4. An overview of the experimentation results for the three FSW builds and policies is shown in Table 10.

Table 10 Experiment Results

FSW Build	Test Policy	Test Results
Basic FSW	**Broad System Policy:** Only legitimate ground stations should command the satellite	**Broad System Policy**: Failure **Note:** malicious ground station gains access
FSW-A	**Policy 1:** Trust Management Event Logging Only; see Table 8	**Broad System Policy:** Failure **Policy 1 Implementation:** Success **Note:** malicious ground station gains access
FSW-A	**Policy 2:** Trust Management Event Logging and Prevention; see Table 8	**Broad System Policy:** Success **Policy 2 Implementation:** Success **Note:** malicious ground station denied access; denial of service experienced
FSW-B	**Policy 3:** Trust Management Event Logging, Prevention and Recovery; see Table 8	**Broad System Policy:** Success **Policy 3 Implementation:** Success **Note:** malicious ground station denied access

In summary, basic FSW takes no specific action to prevent or report malicious activity as described in the commanding abuse case. When the abuse case is applied to the basic FSW, malicious commands are successfully executed when the authentication counter is reached during the forgery attack. This scenario, when presented to FSW-B is identified and processing of the malicious commands is prevented. These results serve to validate the initial hypothesis that trust management principals can be applied to satellite systems to detect and prevent malicious activity.

4.7.1 System Performance.

This section discusses the I-Trust mechanism's performance for detecting the forgety attack abuse case. For satellite commanding operations, signals are transmitted from a ground station through open space to the satellite orbiting above. This transmission path has characteristics which affect transmitted messages. The primary factor to be considered in this research for the command link is the Bit Error Rate (BER).

The command link BER directly affects the number of command messages that are improperly transmitted to the receiver aboard the satellite. This phenomenon affects both malicious and legitimate ground stations resulting in legitimate ground stations occasionally transmitting a command with an invalid command authentication count. Each satellite system in operation has different satellite commanding procedures and command link parameters. Both of these factors contribute to the number of commands received at the satellite with an invalid authentication count.

Due to the nature of satellite commanding in which legitimate commands are lost in transmission, a simple counter for the number of invalid commands received is not directly suitable for security monitoring. The I-Trust mechanism utilized in the CTMS calculates a trust value for entities interacting with the satellite. This value is based upon the quality of interactions relative to an I-Trust marker. The marker for these interactions demonstrated in this work is the command authentication count. As previously discussed, legitimate and malicious ground stations will present commands with invalid authentication counts.

The I-Trust mechanism adjusts the trust value for anonymous entities based the command authentication count field in messages received. An encounter with a malicious entity is flagged when the I-Trust value falls below a specified threshold value. This threshold value, along with the parameters that determine the I-Trust mechanisms operation, must be set according to the specific system in which it is implemented. These settings are determined based upon the number of commands typically lost during a commanding session with a legitimate ground station. Appendix A presents a method and results of optimizing the simple I-Trust algorithm parameters for the command authentication count marker. The performance characteristics for optimal I-Trust parameters are also discussed in Appendix A.

4.7.2 System Characteristics.

The primary goal of demonstrating CTMS architecture in an emulated satellite system environment was to determine the feasibility of implementation in flight ready satellite systems. The characteristics which contribute to the implementations feasibility are Software Compile Size, RAM Utilization, and Function Execution Speed. These characteristics are significant to the implementation of new features into satellite flight software due to a satellites limited hardware resources. A summary of Compile Size and RAM utilization for the Basic FSW and FSW-B builds is shown in Table 11. Additionally, the performance of specific functions which implement the CTMS architecture is shown in terms of execution time in Table 12.

Table 11 FSW Build Characteristics

FSW Build	Constant Data	Code	RAM
FSW-B	2111 Bytes	61443 Bytes	50725 Bytes
Basic FSW	1687 Bytes	51547 Bytes	48600 Bytes
Difference	424 Bytes	9896 Bytes	2125 Bytes

Table 12 CTMS Function Performance

CTMS Function	Execution Time
Simple Interaction	2.4 ms
Policy Evaluation	0.76 ms
Credential Evaluation (AES Decrypt)	21 ms

The compiled characteristics for the FSW builds illustrate the increase in memory usage for an example implementation of the CTMS architecture. Additionally, the function performance shows the added computational time required to process interactions, policies, and credentials with the CTMS implementation. The performance measure taken for the credential evaluation function incorporates the initialization of the AES cryptography function and decryption of the single block of data in the credential.

Each satellite system has unique requirements for hardware and software configurations. Engineers must balance these requirements by making decisions as to which features to implement in the system. Based upon the data presented above, an initial estimate for the system impacts of adding a TMS to a CubeSat mission is realized.

V. Conclusions and Future Work

5.1 Chapter Overview

This chapter presents the conclusions and implications of this thesis. Conclusions for this research focus on the primary thesis question. Additionally, ancillary findings derived from experimentation are discussed. Finally, recommendations for future research are made.

This work developed a multi mechanism Trust Management System (TMS) to address cyber threats to satellite systems. Additionally, methods for threat assessment and vulnerability analysis were presented for use in satellite system development and testing. Chapter I introduced the research problem and focus. Chapter II presented an introduction to the satellite system domain along with computer security and trust management principles. Chapter III covered my approach to the problem and introduced the proposed TMS architecture for satellite telecommanding. Chapter III also covered system security policy, telecommanding abuse case, and satellite Flight Software (FSW) test environment development methodology. Chapter IV presented the test setup used for experimentation and integration of the Consolidated Trust Management System (CTMS) with satellite FSW. Additionally, Chapter IV illustrated the specific forgery attack satellite telecommanding abuse case, and policy used for testing the CTMS implementation. Experiment design, results, and system performance were also presented in Chapter IV.

5.2 Conclusions of Research

This thesis demonstrates the development and use of a TMS to detect the presence of a telecommand forgery attack on satellite FSW. Once detected, the satellite FSW can log or prevent the attack activity. The advantage of using trust management concepts for security in satellite systems is their ability to manage data quality, whereas traditional security mechanisms such as cryptography and access control schemes cannot.

The primary research question of this thesis was to study the application of trust management concepts from the distributed information systems domain to satellite telecommanding. This cross application of research was hypothesized to enhance security in satellite system telecommanding by allowing the detection and denial of adversaries exploiting the command link. This primary research question was broken down into smaller tasks or incremental research questions to fully address the complex nature of the problem.

The first incremental research question was to assess the vulnerability of the basic FSW used as a model in this work. This was accomplished by implementing the basic FSW in the emulated satellite system test environment and applying the forgery attack abuse case. Once the basic FSW was shown to be vulnerable to the forgery attack the effectiveness of the trust management approach could be measured.

The trust management approach addresses the second incremental research question of whether a TMS can be used to detect the forgery attack. This question was addressed by implementing the CTMS architecture in the basic FSW and applying the forgery attack abuse case. Characterization of the FSW with CTMS demonstrated that the system could detect the forgery attack event with a high reliability. The performance of the CMTS architecture can be tuned based upon the environment in which it is operating. A method of selecting tuning parameters for the CTMS architecture was developed and presented in Appendix A for application of the architecture to any specific satellite system. This flexible performance selection feature of the CTMS makes it a robust option for threat detection in satellite FSW.

The third incremental research question in support of this thesis addressed the ability to implement multiple system security policies with the CTMS architecture. The security policies tested addressed the detection and prevention of the telecommanding forgery attack. The policies implemented exercised all components of the CTMS architecture including the Interaction Trust (I-Trust) and credential mechanisms. With the FSW, CTMS implementation, and system security policies configured, the system was tested with the telecommand forgery attack. These tests demonstrated that the CTMS architecture implementation can successfully detect the forgery attack and prevent the execution of malicious commands transmitted by an attacker. As the basic FSW has no inherent user authentication, these malicious commands were prevented by denying all anonymous commands. Notification of the malicious activity and normal system operation was subsequently recovered through the use of a secure command which utilizes the credential trust mechanism for authentication.

The tests performed in this work demonstrates how the CTMS architecture can be used in a satellite system. Through this testing it was shown that the CTMS architecture can be used to consolidate and provide security functions to FSW applications. Additionally, the CTMS architecture has demonstrated potential to be used in conjunction with existing safety and security features found in current satellite systems. An example of these findings is to either merge existing cryptographic authentication with the CTMS architecture as a new trust mechanism, or to simply apply the CTMS architecture in parallel with a dedicated authentication protocol. For the case of a parallel implementation, the authentication protocol would be used to associate entities with trust evidence within the CTMS and also be used for policy evaluation.

Results from the incremental research questions prove the hypothesis that trust management principles may be applied to satellite system telecommanding to enhance security. The work completed in this thesis is the demonstration of a powerful new satellite security methodology and tool. This approach can not only be used to protect satellites from the specific forgery attack case presented here, however may be applied as a method to protect satellites from a wide range of threats.

5.3 Recommendations for Future Research

This work consisted of an effort to bring trust management practices from the distributed information systems and computer security domain to satellite system FSW. The concepts presented in this thesis can be extended in several ways.

First, further identification and characterization of satellite system abuse cases will benefit work towards securing these systems. New abuse cases can then be applied to satellite systems to identify vulnerabilities, which can then be addressed with trust management architectures such as CTMS. Second, further characterization of the I-Trust calculation methods can be performed to better understand how to apply the algorithms to solve security problems in satellite systems. Finally, a more detailed implementation of

CTMS could be applied to a FSW with authentication and encryption to further demonstrate the TMS capabilities in supporting these traditional security mechanisms.

Appendix A. Simple I-Trust Algorithm Optimization

The command authentication count used in a satellite telecommanding architecture is primarily a safety feature. The function of the command authentication count is to identify commands which are received out of sequence. The typical response to receiving a command with an invalid authentication count is to discard the message. This response inadvertently addresses commands sent with malicious intent. A third party whom is unaware of the current command authentication count in the satellite will be unable to transmit valid commands to be processed by the satellite. This indicates a potential security benefit of the command authentication count feature.

In order to evaluate the command authentication count for use as a security mechanism, an analysis of the feature's properties is performed. The first issue addressed is the probability an attacker will accurately guess the authentication count. Second is how to detect an attacker attempting to access the system by guessing the authentication count. Lastly, an analysis of the Interaction Trust (I-Trust) mechanism and optimization of the I-Trust configuration parameters is presented.

The probability an attacker will guess the authentication count is directly related to the range of values for the authentication count field. The authentication count used as a demonstration implementation in this work is a 16-bit variable. This results in 2^{16} possible values with a maximum of 65,535 and a minimum of 0. An adversary attempting to guess this value has a one in 65,536 chance to succeed on the first try. Using the forgery attack scenario an attacker will choose a value only once and the probability of correctly selecting the authentication count is computed with Equation 1.

The number of attempts to guess the authentication count is n and the probability of success will increase with each attempt. At this rate the attacker must try 656 times to have approximately 1% chance of guessing the authentication count.

$$P = \sum_{x=0}^{n-1} \frac{1}{2^{16} - x}$$

Equation 1 Probability

An attacker attempting to access a system utilizing an authentication counter must first acquire the current authentication count. One attack scenario presented in this work involves the attacker sending multiple commands with an incrementally different authentication count for each. This method of starting from an initial value for the authentication count and making a series of attempts with sequential authentication count values allows the attacker to cover all values of the authentication counter and access the satellite.

Abuse of the command authentication with this activity leaves evidence in the satellite system. This evidence is the pattern and history of commands received by the system with an invalid authentication count. The evidence will appear differently depending on whether this attempt is made independent of a legitimate ground station's telecommanding session or during a legitimate telecommanding session.

Failure to present the proper authentication count can indicate either indicate lost legitimate commands or the presence of an attacker attempting illegitimate commands. In order to differentiate between legitimate ground station and attacker in a commanding encounter, these authentication failures are aggregated with the use of an I-Trust algorithm. This algorithm computes a value which is an indicator of the reliability or trustworthiness of a remote system. The algorithm computes the trust value based upon an interaction's modification of a specific marker in the system. If the marker indicates cooperation by the remote system then the interaction trust value is increased. Conversely, if the marker indicates defection by the remote system the interaction trust value is reduced. This system of computing a trust value which incorporates the outcome of interactions is used to detect the presence of an attacker in the system.

The particular algorithm used to perform this trust value calculation is adapted from the agent rating process presented by Yu and Sing. This algorithm utilizes two parameters to define how much the trust value should increase or decrease following an interaction. The environment in which this algorithm is being used will have an effect on how these parameters should be set. The remainder of this Appendix documents the characterization of the satellite system telecommanding environment relevant to the command authentication counter. Additionally, a procedure for optimizing the algorithm parameters used to detect attack behavior in the command system is presented.

The optimization and characterization of the simple I-Trust algorithm is presented in the context of malicious activity detection in satellite system telecommanding. The algorithm parameter optimization is presented in steps to configure the I-Trust algorithm for a particular satellite system. This procedure was developed after the analysis and optimization of the algorithm's parameters α and β for a specific satellite system.

The first step to configure the I-Trust algorithm is to establish a desired false negative threshold. This threshold is based upon the users tolerance for potential false negatives in the system. For this example, a value of 0.001 is chosen. This indicates that the user will accept at the very most a one in one thousand chance an attacker will succeed unnoticed. This value is used to compute the approximate maximum number of command attempts which are made during an attack on the system before being detected.

The calculation is performed by multiplying the probability of success by the number of possible values in the command authentication counter. For my test system, the command authentication counter is 16 bits, which results in 65,536 possible values. See Figure 8 Attempt Approximation Formula for the approximation formula and example computation. Alternatively, the user may pick the number of attempts an attacker may make on the system and compute the false negative probability.

$$\boxed{\begin{aligned} &P(x) = n\frac{1}{2^b} \\ &n = P(x) * 2^b \\ &n = (.001) * 65{,}536 \\ &n \approx 65 \end{aligned}}$$

Figure 8 Attempt Approximation Formula

The second step in configuring optimal parameters for the I-Trust algorithm is to establish a system security policy, which sets a threshold on the I-Trust value. This policy limit value is also related to the security posture of the system. A high value (less negative) will result in a sensitive system, which is in turn more susceptible to false positives. A low value will result in a system which requires higher penalties to reach the policy limit. This is due to the requirement to identify malicious behavior within the number of defection interactions calculated in step one.

The value of -0.8 was chosen and is used here as it falls just below the approximately linear portion of the trust curve for a series of defection interactions. This is best illustrated by the attack curve in Figure 17. A policy limit at the end of this approximately linear portion of the curve allows the system to accommodate both a high security posture and low false positive rate.

With the chosen security policy limit on the I-Trust value, the third step to configure the I-Trust mechanism is to establish a bound for the value β. This is done by computing the final trust value after 65 defections with consecutively decreasing values of β. The iteration that results in a final trust value less than or equal to the policy limit value, e.g., -0.8 is the upper bound on the β parameter. No α value is required to compute this bound for β as all of the interactions used in the calculation are defection. This method is also described by the pseudo code in Figure 9.

```
for(Beta = -.0001; Beta >= -.5; Beta = Beta - .0001){
      T = 0; // I-Trust Variable
      for(count = 1; count <= MAX_ATTEMPTS; count++){
            Simple_Interaction(Defection); // Computes I-Trust with
            Beta
      }
      if (T <= POLICY_LIMIT){
            return Beta;
      }
}
```

Figure 9 Beta Bound Pseudo Code

The fourth step is to determine additional bounds for α and β. These bounds are used to reduce the search space required to establish the optimal I-Trust parameters for a specific system. With the upper bound for β computed previously, the lower bound for β along with bounds for α are established.

The lower bound on β is based upon the nature of the modeled behavior and the design of the I-Trust algorithm. This lower bound for β is set at -0.5 for this system optimization, as higher values would be unrealistic for this. As the parameter beta approaches -0.5, the parameter α must also increase in magnitude to avoid false positives in the system. Values for β above -0.5 do not provide effective results when modeling trust based on the command authentication count. This is due to instability associated with large β values and the associated large α values required to maintain low false positive rates. The bounds for β as discussed are shown on a number line representing the search space for optimal α and β values, see Figure 10.

With both bounds for β established, the bounds for optimal α are addressed. The upper bound for α is limited by the absolute value of β. Additionally, the α parameter has a lower bound of zero. The zero α lower bound is a result of the positive nature of the α parameter discussed in Section 3.3.3.1. These relationships are also captured in Figure 3, which highlights the bounds and exclusion areas for α and β. The exclusion areas are marked by hashed boxes, which indicate values not included in the parameter search. Horizontal arrows on the number line indicate the direction of the parameter search.

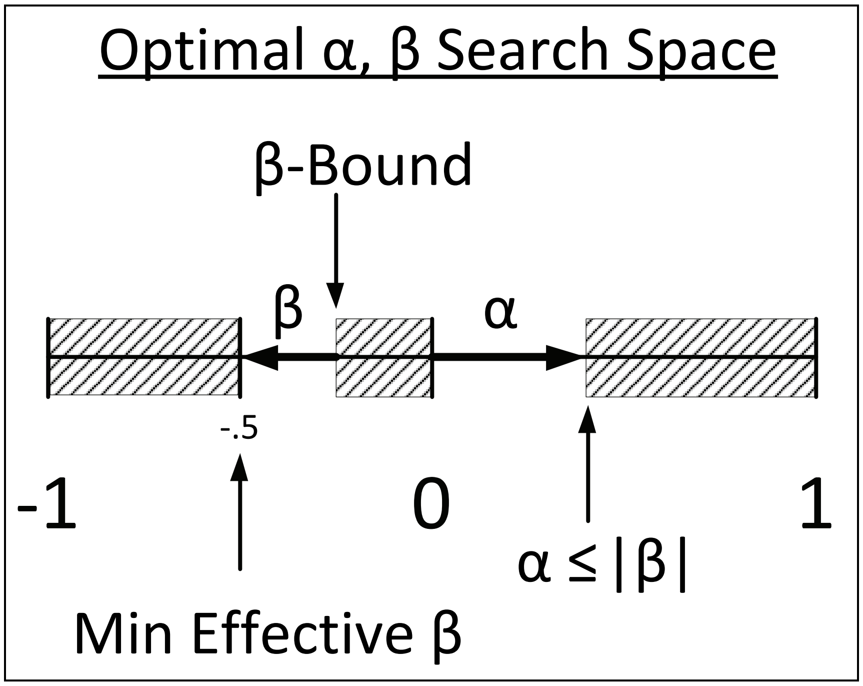


Figure 10 Number Line for Optimizing α and β Parameters

The final step in configuring the I-Trust mechanism is to search the possible values of α and β that provides the fewest false positives, and meets the desired security posture. Searching the available values of α and β requires additional information regarding the system being configured. The critical factors based upon system design necessary to perform α, β optimization are the expected failure (defection) rate for a legitimate user and the number of interactions per encounter. Additionally, the configurable search parameters are: the step size for incrementing α and β, the number of times to sample each random encounter, the desired maximum false positive rate for the system, and bounds for average trust value. The average trust value relates directly to the desired system security posture by keeping I-Trust values balanced which enables reliable detection of an attack during a command encounter.

The search method used to identify the optimal α and β values begins with a loop over the β value starting at the previously established upper bound. This β loop will run a second loop which will search α from zero to the absolute value of the current β. These nested loops will cover the bounded values for both I-Trust algorithm parameters, see Figure 11.

```
for(Beta = Beta_Bound; Beta >= -.5; Beta = Beta - BETA_INCREMENT){
      for(Alpha = 0; Alpha <= absval(Beta); Alpha = Alpha +
ALPHA_INCREMENT){
            Series_Of_Encounters();
      }
}
```

Figure 11 α and β Loop Pseudo Code

Within the α loop, a series of encounters are executed base upon the number established by the search parameter. This series of encounters is a loop over the number of encounters which computes each series of interactions. The number of encounters computed should be large enough to significantly indicate the average number of false positives the system will generate per encounter. The average interaction trust value for each encounter is computed, along with the average of all encounters for a given set of parameters (α, β). Additionally, the number of false positives experienced during the series of encounters is calculated. Pseudo code for the series of encounters loop is shown in Figure 12.

```
Series_Of_Encounters(){
      for(Encounter = 0; Encounter < RANDOM_ENCOUNTERS; Encounter++){
            Interaction_Series();
            if (ISeries_False_Positive > 1){
                  Encounter_False_Positive++;
            }
            ISeries_Avg_Sum = ISeries_Avg_Sum + ISeries_Avg;
      }
      Encounter_Trust_Avg = ISeries_Avg_Sum / RANDOM_ENCOUNTERS;
}
```

Figure 12 Series of Encounters Pseudo Code

The series of encounters loop contains a loop to execute a series of interactions in an encounter. This interaction series loop processes the number of interactions specified for each encounter, while calculating statistics necessary for the series of encounters. Each interaction is determined to be either cooperation or defection based upon the system being modeled. In this case each legitimate interaction has a one in ten chance of being defection. If at any time in the interaction series the trust value falls below the policy limit, a false positive is counted. A pseudo code example of this loop is shown in Figure 13.

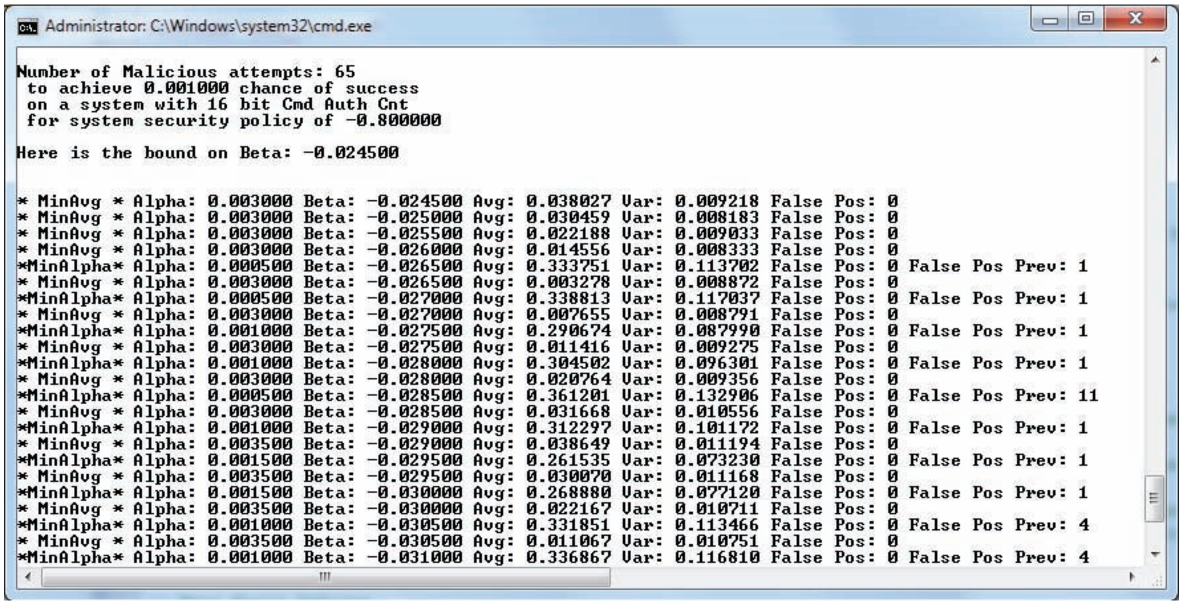
```
Interaction_Series(){
      //All variables are initialized to zero
      for (Interaction = 1; Interaction <= NUM_INTERACTIONS;
Interaction++){
            if ((rand() % 10 + 1) == 1) IR=0;
            else IR=1;
            // Run the random interaciton through the Simple Trust
Algorithm
            // Alpha and Beta are set by their loops
            // An interaction of 0 is defection; 1 is cooperation
            Simple_Interaction(IR);
            // Check for false positive in interaction set
            if(T <= POLICY_LIMIT){
                  ISeries_False_Pos ++ ;
            }
            //Accumulate a sum to average the Interaction Trust Values
            ITrust_Sum = ITrust_Sum + T;
      }
      // Calculate the average interaction trust value
      ISeries_Avg = ITrust_Sum / NUM_INTERACTIONS;
}
```

Figure 13 Interaction Series Pseudo Code

The search for optimal I-Trust parameters is complete when user conditions are met with regards to false positives and average trust value. These parameters are checked at the end of each series of encounters and are reported as the result of the optimization for α and β. The initial values returned from the search are optimized based upon the input parameters. The results of this process is illustrated with a complete example; see Table 13. A sample screen shot from the optimization tool which calculated the optimal I-Trust parameters is shown in Figure 14.

Table 13 Optimization Example 1

Optimization Example Setup		Optimization Example Results	
False Negative Rate	.001	Attempts	65
Policy Limit	-.8	β Bound	-.0245
Step Size	.0005	α Result	.0025
False Positive Rate	0	β Result	-.0245
Average Trust Value	±.06	Result Avg Trust	-.04
Expected Failure Rate	1/10	False Positives	0
Interactions Per Encounter	200		
Encounter Samples	1,000		



```
Number of Malicious attempts: 65
 to achieve 0.001000 chance of success
 on a system with 16 bit Cmd Auth Cnt
 for system security policy of -0.800000

Here is the bound on Beta: -0.024500

* MinAvg * Alpha: 0.003000 Beta: -0.024500 Avg: 0.038027 Var: 0.009218 False Pos: 0
* MinAvg * Alpha: 0.003000 Beta: -0.025000 Avg: 0.030459 Var: 0.008183 False Pos: 0
* MinAvg * Alpha: 0.003000 Beta: -0.025500 Avg: 0.022188 Var: 0.009033 False Pos: 0
* MinAvg * Alpha: 0.003000 Beta: -0.026000 Avg: 0.014556 Var: 0.008333 False Pos: 0
*MinAlpha* Alpha: 0.000500 Beta: -0.026500 Avg: 0.333751 Var: 0.113702 False Pos: 0 False Pos Prev: 1
* MinAvg * Alpha: 0.003000 Beta: -0.026500 Avg: 0.003278 Var: 0.008872 False Pos: 0
*MinAlpha* Alpha: 0.000500 Beta: -0.027000 Avg: 0.338813 Var: 0.117037 False Pos: 0 False Pos Prev: 1
* MinAvg * Alpha: 0.003000 Beta: -0.027000 Avg: 0.007655 Var: 0.008791 False Pos: 0
*MinAlpha* Alpha: 0.001000 Beta: -0.027500 Avg: 0.290674 Var: 0.087990 False Pos: 0 False Pos Prev: 1
* MinAvg * Alpha: 0.003000 Beta: -0.027500 Avg: 0.011416 Var: 0.009275 False Pos: 0
*MinAlpha* Alpha: 0.001000 Beta: -0.028000 Avg: 0.304502 Var: 0.096301 False Pos: 0 False Pos Prev: 1
* MinAvg * Alpha: 0.003000 Beta: -0.028000 Avg: 0.020764 Var: 0.009356 False Pos: 0
*MinAlpha* Alpha: 0.000500 Beta: -0.028500 Avg: 0.361201 Var: 0.132906 False Pos: 0 False Pos Prev: 11
* MinAvg * Alpha: 0.003000 Beta: -0.028500 Avg: 0.031668 Var: 0.010556 False Pos: 0
*MinAlpha* Alpha: 0.001000 Beta: -0.029000 Avg: 0.312297 Var: 0.101172 False Pos: 0 False Pos Prev: 1
* MinAvg * Alpha: 0.003500 Beta: -0.029000 Avg: 0.038649 Var: 0.011194 False Pos: 0
*MinAlpha* Alpha: 0.001500 Beta: -0.029500 Avg: 0.261535 Var: 0.073230 False Pos: 0 False Pos Prev: 1
* MinAvg * Alpha: 0.003500 Beta: -0.029500 Avg: 0.030070 Var: 0.011168 False Pos: 0
*MinAlpha* Alpha: 0.001500 Beta: -0.030000 Avg: 0.268880 Var: 0.077120 False Pos: 0 False Pos Prev: 1
* MinAvg * Alpha: 0.003500 Beta: -0.030000 Avg: 0.022167 Var: 0.010711 False Pos: 0
*MinAlpha* Alpha: 0.001000 Beta: -0.030500 Avg: 0.331851 Var: 0.113466 False Pos: 0 False Pos Prev: 4
* MinAvg * Alpha: 0.003500 Beta: -0.030500 Avg: 0.011067 Var: 0.010751 False Pos: 0
*MinAlpha* Alpha: 0.001000 Beta: -0.031000 Avg: 0.336867 Var: 0.116810 False Pos: 0 False Pos Prev: 4
```

Figure 14 I-Trust Parameter Optimization Tool

With an input of 0.001 for an acceptable false negative rate the resulting number of attempts is 65. The policy limit of -0.8 establishes an initial bound for β of (-0.0245). The initial result is α = 0.0025, β = -0.0245, which meets the requirements for false positives and average trust rating. A graph displaying this result for Example 1 is shown in Figure 15. From the graph of the interaction trust value versus interaction number we see the trust value drops below the policy limit exactly at the required 65 interactions for the initial abuse case (where the red line crosses -0.8). Additionally, the legitimate user will maintain an average trust rating of 0.04.

A second abuse case is also shown where the attacker transmits commands to be processed during the legitimate ground station's command session. This activity begins at interaction 50 and continues through the end of the simulation. The I-Trust value for this case drops below the policy limit before the end of the encounter, however it requires more defection interactions to reach this limit. This is due to the occasional cooperation interactions supplied by the legitimate ground station.

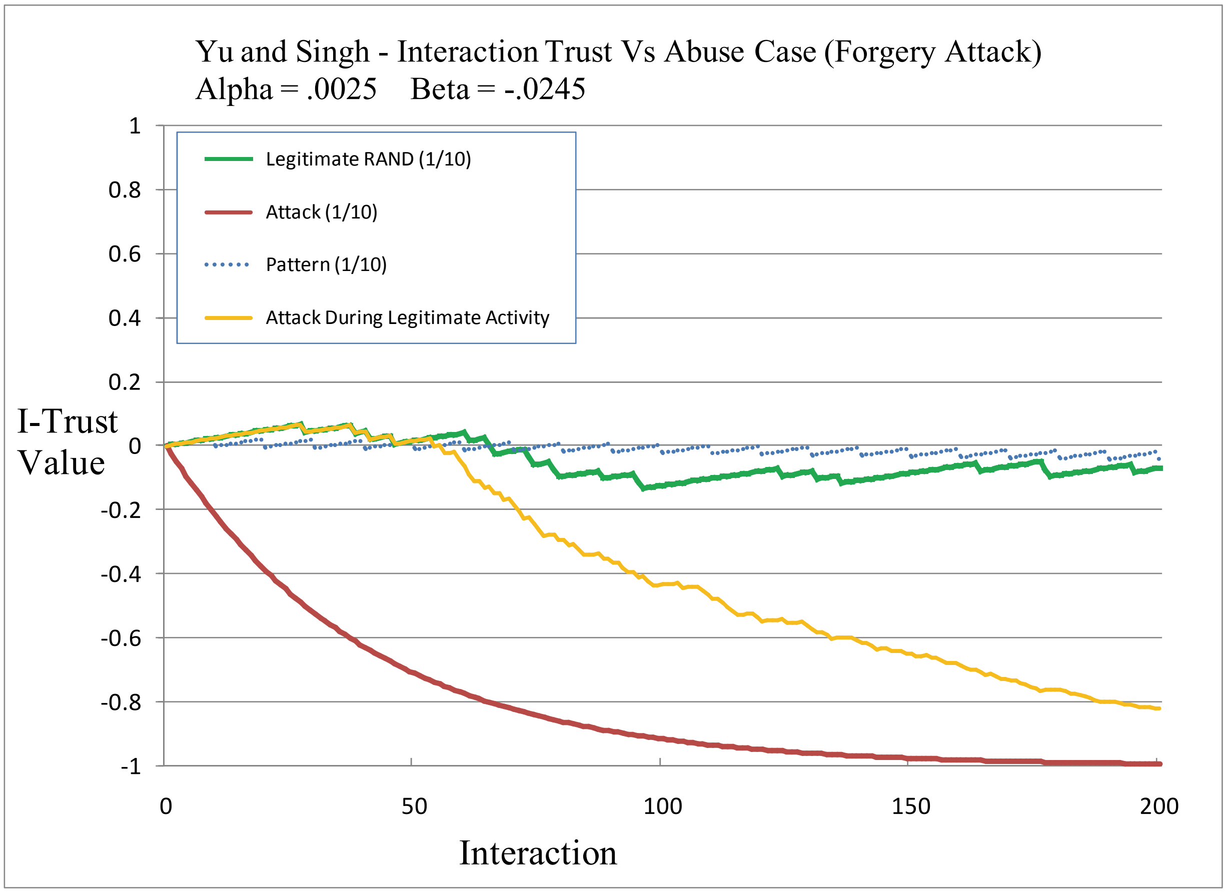


Figure 15 Example 1: Initial I-Trust Optimization

While the α and β values in the Example 1 result are optimal for the input requirements, they can be enhanced for faster response to the malicious events while maintaining a low false positive rate. This is achieved by continuing the search process through α, β and selecting a set of parameters which results in zero false positives after an increment in β which results in extensive false positives. This selection of I-Trust parameters is made without consideration for the trust average. The resulting parameters are taken at the point where the minimum α is given for the current β, while maintaining a user acceptable false positive rate. By selecting this solution we get increased security potential with minimal false positives. An example which illustrates this solution is shown in Figure 16, with optimization settings in Table 14.

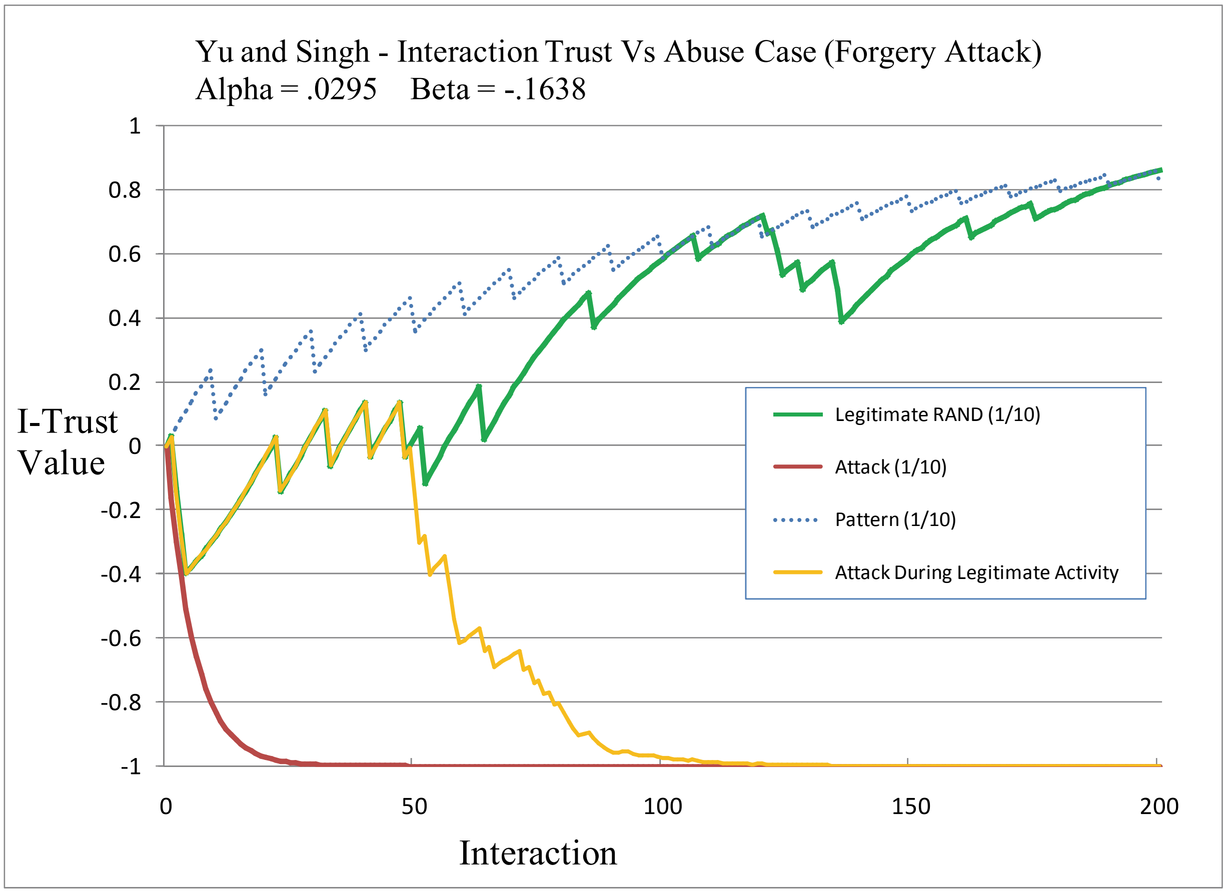


Figure 16 Optimization Example 2: No Average Constraint, Min Alpha for Beta

Table 14 Optimization Example 2

Optimization Example Setup		**Optimization Example Results**	
False Negative Rate	.00015	Attempts	9
Policy Limit	-.8	β Bound	N/A
Step Size	.0005	α Result	.0295
False Positive Rate	0	β Result	-.1638
Average Trust Value	N/A	Result Avg Trust	.675
Expected Failure Rate	1/10	False Positives	0
Interactions Per Encounter	200		
Encounter Samples	1,000		

Example 2 shows increased performance for identifying the initial attack pattern, with 9 attempts bringing the trust value down to the policy limit. Additionally, the number of false positives for the series of encounters is zero, which is the same as in Example 1. This modified method of selecting α and β results in a higher overall trust average of 0.675, which may not be suitable when identifying attacks during extended encounters. An example of where this optimization would not fit a mission requirement is where emphasis is placed on the threat of an attacker intruding on an ongoing command session. With reference to Figure 16, if an attacker were to begin the forgery attack after legitimate interaction 150 instead of 50 as shown, the system would require additional defection interactions to identify the attack. This results in a final method for selecting optimal α and β parameters.

Both methods for optimizing selecting I-Trust configuration parameters are combined, which will result in: minimum α for the current β, an overall average trust rating within a specified range, and false positives within a user defined range. Results from this selection method can identify an attack independent of an active legitimate commanding session in fewer interactions than the initial method. Additionally, this selection method provides an active defense posture during a legitimate commanding session not seen with the first two selection methods. An example of optimization results utilizing this selection method are shown in Table 15 and Figure 17.

Table 15 Optimization Example 3

Optimization Example Setup		Optimization Example Results	
False Negative Rate	.00035	Attempts	22
Policy Limit	-.8	β Bound	N/A
Step Size	.0005	α Result	.0095
False Positive Rate	0	β Result	-.0789
Average Trust Value	±.06	Result Avg Trust	.059
Expected Failure Rate	1/10	False Positives	0
Interactions Per Encounter	200		
Encounter Samples	1,000		

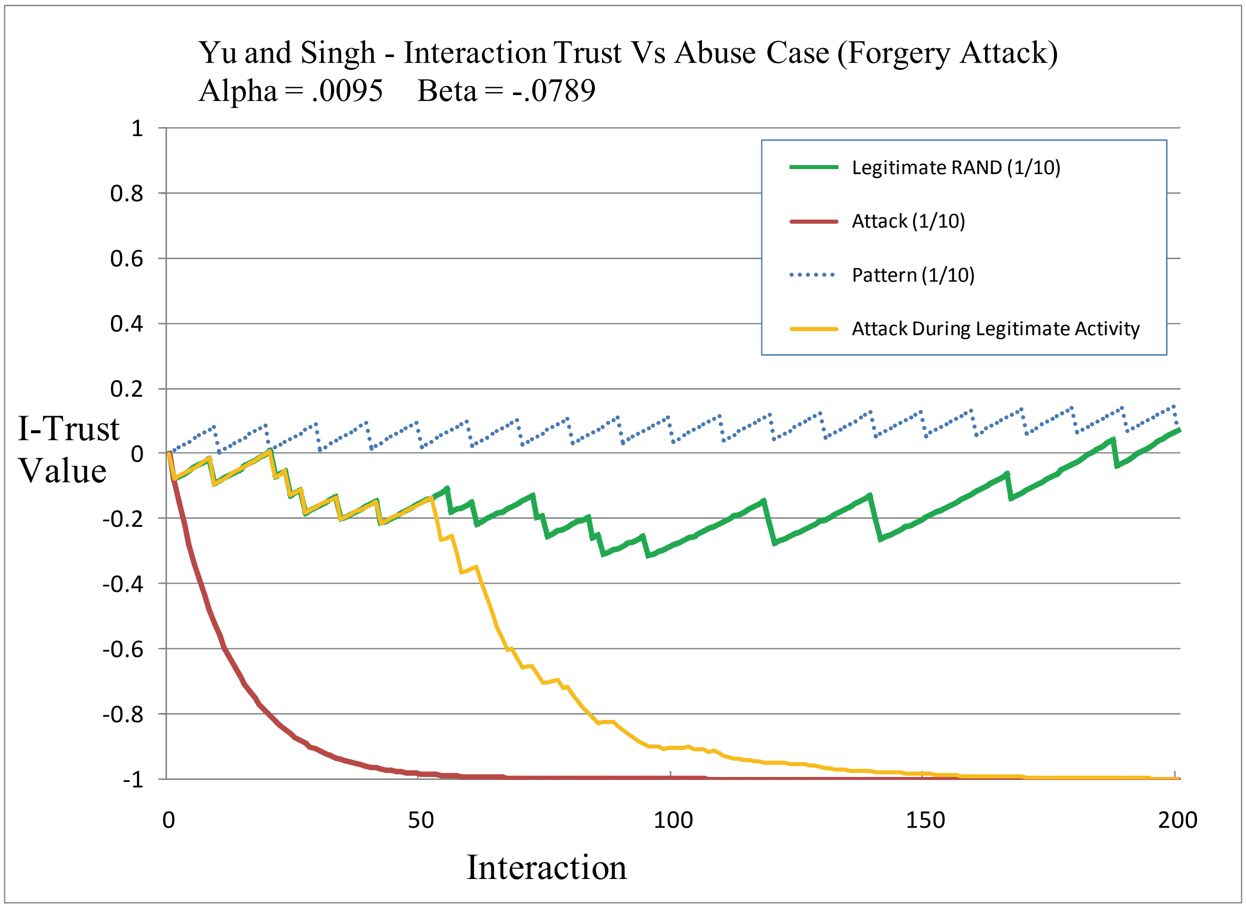


Figure 17 Optimization Example 3: Average Constraint, Minimum Alpha for Beta

Example 3 shows the result of the optimization incorporating the minimum α for β, constrained average trust rating, and constrained false positives. The key benefit of this method is a compromise between initial security and extended defense posture. The low absolute value of average trust rating provides relatively constant threat response throughout a command encounter. False positives are also managed with this method with zero false positives reported in 1,000 command encounter samples.

The optimization results with constrained average, false positives , and minimum α can only exist within a small band of possible parameter values. These represent the globally optimum results and are acquired by searching the parameter space starting from high values of β. Figure 18 illustrates pseudo code implementing the three methods of selecting and reporting results from the parameter optimization.

```
if (Encounter_Trust_Avg > -1 * AVG_TRUST && Encounter_Trust_Avg <
AVG_TRUST){
      //Extract Result "Initial Optimization - Average in range"
{

if((false_pos_sum < false_pos_prev)&&(false_pos_sum <=
FALSE_POS_RATE)){
      if ((Encounter_Trust_Avg > -1 * AVG_TRUST) &&
(Encounter_Trust_Avg < AVG_TRUST)){
          //Extract Result "Average in range, Minimum Alpha for Current
Beta"
      {
      else{
          //Extract Result "Minimum Alpha for Current Beta"
      }
}
false_pos_prev = false_pos_sum;
```

Figure 18 Optimization Result Selection

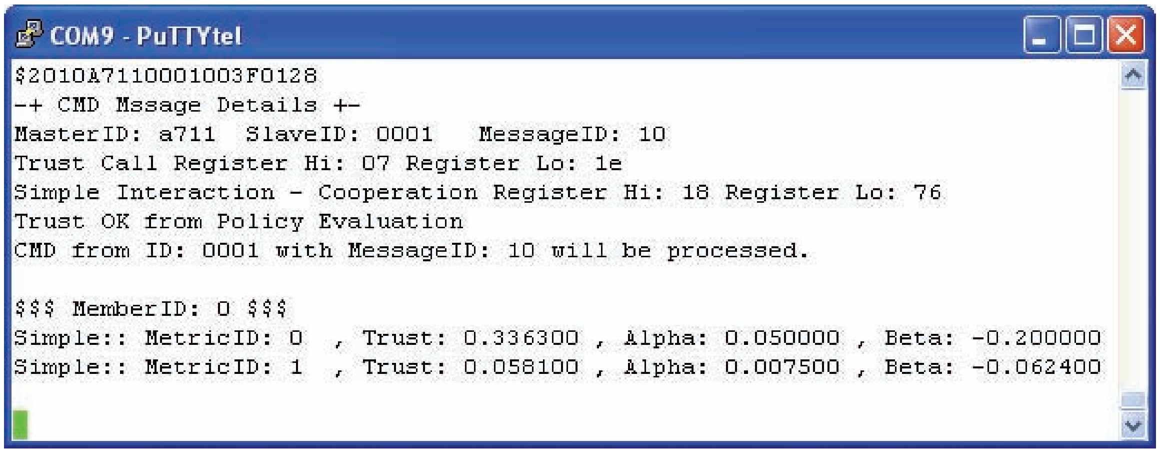
Appendix B. Illustrated CTMS Test Sequence

Utilizing the test environment described in Section 4.2, three Flight Software (FSW) builds were tested with the forgery attack sequence presented in Section 4.4. The FSW build used in the following illustrations is FSW-B, which consisted of basic CubeSat FSW with the Consolidated Trust Management System (CTMS) architecture implementation. The FSW was also modified to utilize the CTMS implementation by inserting CTMS Application Programming Interface (API) calls into the command handler application to monitor command interactions and enforce policy actions. Furthermore, a CTMS specific telecommand (secure unlock) was added to the basic flight software which utilizes the credential trust mechanism to authenticate ground stations. The CTMS secure unlock command was used in this test scenario to acknowledge the detection of an attack sequence and to restore commanding functionality for anonymous ground stations. This is necessary as anonymous commanding can be disabled by system security Policy 3 described in Section 4.5.

The first step in this example CTMS test sequence corresponds to a normal satellite telecommanding scenario. The legitimate ground station in this step transmits a sequence of commands without user authentication to the satellite. This activity increments the satellite's onboard command authentication counter and increases the Interaction Trust (I-Trust) value for the anonymous CTMS user.

The satellite diagnostic port is monitored during the legitimate command sequence. The diagnostic port displays the following information: the byte pattern for each command received, command specific details, a policy evaluation result, and the current CTMS I-Trust data. Figure 19 is the last message displayed on the satellite diagnostic port after the legitimate command sequence is completed.

The first line in the diagnostic output shown in Figure 19 is the byte sequence for the last telecommand received. The second line is a header indicating the following five lines are details regarding the processing of the last command received. These details indicate what type of message was received (MessageID), the result of a policy check (Trust OK...), and the action following the policy check (CMD... will be processed). The remaining information is the current CTMS I-Trust data. This data indicates that the entity tracked in the TMS with MemberID 0 has two associated simple I-Trust metrics. MemberID 0 represents interactions from unauthenticated users. The I-Trust metrics correspond to trust values computed based upon the command authentication count marker using the simple I-Trust algorithm. The reason for computing multiple trust values for the same marker is to demonstrate the configurable nature of the CTMS architecture to match a target systems specific performance profile. The topic of performance is addressed further in Section 4.7.1 and Appendix A.



```
$2010A7110001003F0128
-+ CMD Mssage Details +-
MasterID: a711  SlaveID: 0001   MessageID: 10
Trust Call Register Hi: 07 Register Lo: 1e
Simple Interaction - Cooperation Register Hi: 18 Register Lo: 76
Trust OK from Policy Evaluation
CMD from ID: 0001 with MessageID: 10 will be processed.

$$$ MemberID: 0 $$$
Simple:: MetricID: 0  , Trust: 0.336300 , Alpha: 0.050000 , Beta: -0.200000
Simple:: MetricID: 1  , Trust: 0.058100 , Alpha: 0.007500 , Beta: -0.062400
```

Figure 19 Step 1, Satellite Diagnostic Port Output

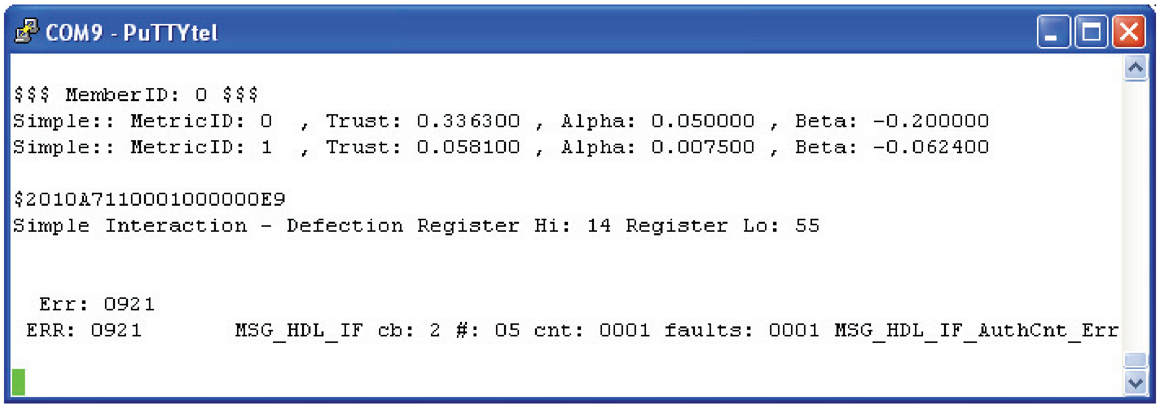
The I-Trust values displayed for Metric 0 and Metric 1 are positive values, which indicate the series of legitimate commands received during the normal command sequence. These values were both initialized to zero and have different values in Figure 19 due to each metric being associated with individual I-Trust parameters as shown. MetricID 0 is used in the policy evaluation for this experiment, and as the trust value for this metric is above the policy limit of -0.5 the command was processed.

The second step in the CTMS test is the execution of a forgery attack on the satellite. This attack is an instance of the abuse case described in Section 4.4. Once the legitimate command session has completed and the satellite has moved into the attackers field of view, the malicious sequence of commands are transmitted.

The malicious commands in this experiment are transmitted by a custom commanding tool. These commands are similar to those which were sent by CGA in Step 1, however the proper command authentication count is unknown to the attacker. Figure 20 shows the custom commanding tool setup for this step of the experiment.

The command selected for transmission is a simple no operation command. This commands only function is to test the command transmission and execution path. Below the command selection in Figure 20 are the command header fields, which are setup for the no operation command. There are no arguments for this command so the command data field is blank. The Command Authentication Count (Auth Count) field is shaded to indicate that for the forgery attack this field will be incremented after transmission for the number of attempts indicated.

Figure 20 Step 2, Custom Commanding Tool Setup



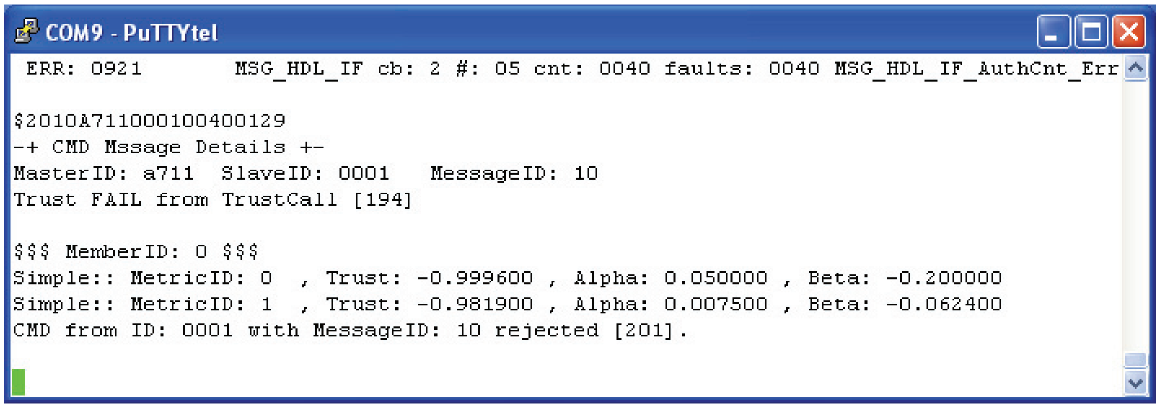
COM9 - PuTTYtel

```
$$$ MemberID: 0 $$$
Simple:: MetricID: 0  , Trust: 0.336300 , Alpha: 0.050000 , Beta: -0.200000
Simple:: MetricID: 1  , Trust: 0.058100 , Alpha: 0.007500 , Beta: -0.062400

$2010A7110001000000E9
Simple Interaction - Defection Register Hi: 14 Register Lo: 55


  Err: 0921
 ERR: 0921        MSG_HDL_IF cb: 2 #: 05 cnt: 0001 faults: 0001 MSG_HDL_IF_AuthCnt_Err
```

Figure 21 Step 2, Invalid Command Authentication Count Error

During the attack, telecommands are received by the satellite with an invalid command authentication count value. The error generated by this activity can be seen on the diagnostic port, see Figure 21. Portions of the diagnostic output from the last legitimate command processed remains on the screen while the attackers command and error information are below. Each command received with an invalid command authentication count is considered a defection by an anonymous ground station and the I-Trust value for the CTMS member ID is decreased.

Once the attack sequence reaches the current satellite command authentication count, the attackers command is evaluated based upon system security policy. The trust value shown in the diagnostic output for this event is now -0.9996, which is well below the policy limit of -0.5, see Figure 22. The output also shows the policy evaluation, which resulted in the command being rejected.



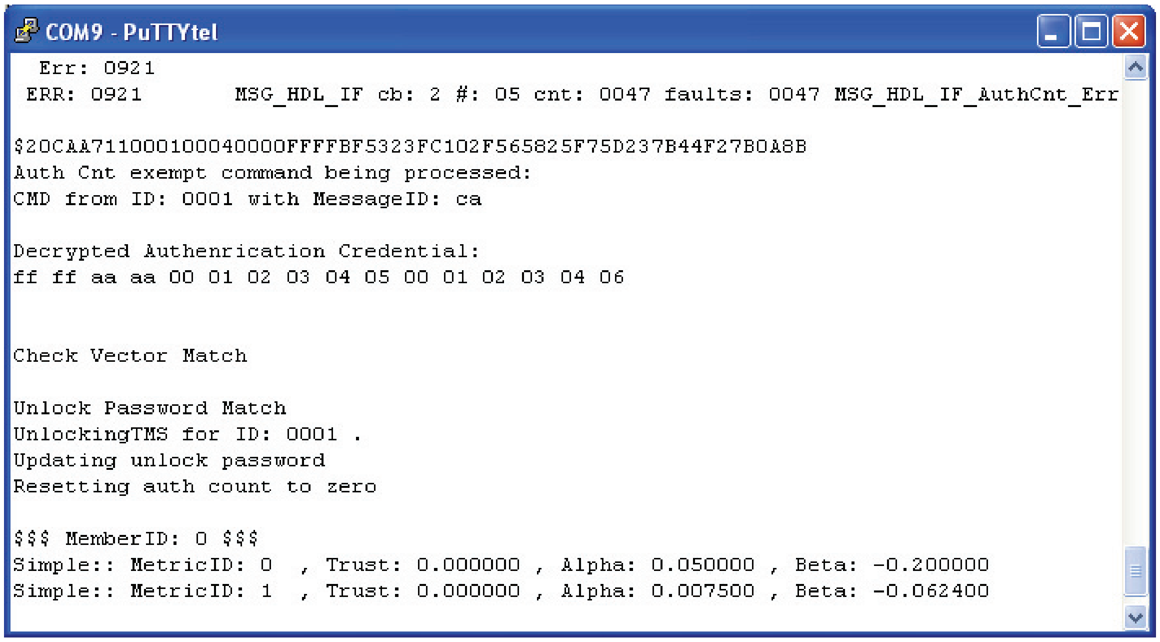
```
 ERR: 0921       MSG_HDL_IF cb: 2 #: 05 cnt: 0040 faults: 0040 MSG_HDL_IF_AuthCnt_Err

$2010A711000100400129
-+ CMD Mssage Details +-
MasterID: a711  SlaveID: 0001   MessageID: 10
Trust FAIL from TrustCall [194]

$$$ MemberID: 0 $$$
Simple:: MetricID: 0  , Trust: -0.999600 , Alpha: 0.050000 , Beta: -0.200000
Simple:: MetricID: 1  , Trust: -0.981900 , Alpha: 0.007500 , Beta: -0.062400
CMD from ID: 0001 with MessageID: 10 rejected [201].
```

Figure 22 Step 2, Trust Policy Check During Attack

As the interaction trust values for anonymous entities is now below the policy limit it can only be restored through the secure unlock command. This command is generated and transmitted to the satellite using the custom commanding tool, see Figure 23. The secure unlock command contains a unique authentication credential as an argument in the command data field. Upon processing this command, the satellite will restore the I-Trust values to zero which will re-enable anonymous commanding.

Figure 23 Secure Unlock Command Transmission

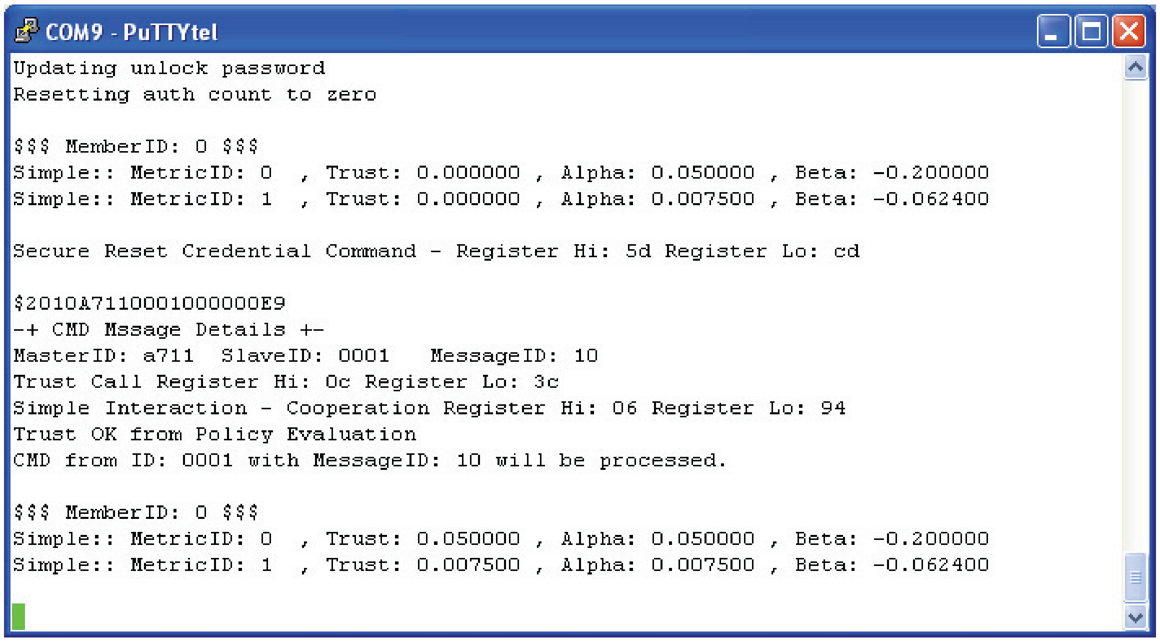
The third CTMS test step is execution of the secure unlock command. The unlock command is processed to evaluate the authentication credential, which is a more authoritative form of trust than the interaction trust value. The diagnostic output for the credential evaluation follows the verification procedure outlined in Section 4.3.2. This unlock command follows the last invalid command sent by the malicious user, which is indicated by the error at the top of the diagnostic display, see Figure 24. The decrypted authentication credential is displayed on the diagnostic port followed by the checks necessary to validate the credential. All of the checks are successful in this test which results in the I-Trust trust values for the anonymous MemberID being set to zero along with the command authentication counter.



```
COM9 - PuTTYtel
  Err: 0921
 ERR: 0921       MSG_HDL_IF cb: 2 #: 05 cnt: 0047 faults: 0047 MSG_HDL_IF_AuthCnt_Err

$20CAA7110001000400000FFFFBF5323FC102F565825F75D237B44F27B0A8B
Auth Cnt exempt command being processed:
CMD from ID: 0001 with MessageID: ca

Decrypted Authenrication Credential:
ff ff aa aa 00 01 02 03 04 05 00 01 02 03 04 06


Check Vector Match

Unlock Password Match
UnlockingTMS for ID: 0001 .
Updating unlock password
Resetting auth count to zero

$$$ MemberID: 0 $$$
Simple:: MetricID: 0  , Trust: 0.000000 , Alpha: 0.050000 , Beta: -0.200000
Simple:: MetricID: 1  , Trust: 0.000000 , Alpha: 0.007500 , Beta: -0.062400
```

Figure 24 Step 3, Secure Unlock Command Processing

The final step in this example CTMS test is to transmit legitimate anonymous commands to the satellite for processing. Since the secure unlock command was executed, the I-Trust value for anonymous commanding is set to zero. Since the I-Trust value is now greater than the policy limit of -0.5, anonymous commands will be processed. Additionally, the command authentication counter was reset to zero, which will allow the legitimate ground station to begin commanding with that count. The first successful legitimate command after the unlock is verified with the diagnostic port output shown in Figure 25.


COM9 - PuTTYtel

```
Updating unlock password
Resetting auth count to zero

$$$ MemberID: 0 $$$
Simple:: MetricID: 0  , Trust: 0.000000 , Alpha: 0.050000 , Beta: -0.200000
Simple:: MetricID: 1  , Trust: 0.000000 , Alpha: 0.007500 , Beta: -0.062400

Secure Reset Credential Command - Register Hi: 5d Register Lo: cd

$2010A7110001000000E9
-+ CMD Mssage Details +-
MasterID: a711  SlaveID: 0001   MessageID: 10
Trust Call Register Hi: 0c Register Lo: 3c
Simple Interaction - Cooperation Register Hi: 06 Register Lo: 94
Trust OK from Policy Evaluation
CMD from ID: 0001 with MessageID: 10 will be processed.

$$$ MemberID: 0 $$$
Simple:: MetricID: 0  , Trust: 0.050000 , Alpha: 0.050000 , Beta: -0.200000
Simple:: MetricID: 1  , Trust: 0.007500 , Alpha: 0.007500 , Beta: -0.062400
```

Figure 25 Step 4, Legitimate Commanding Following Unlock

Bibliography

[1] US Congress, Uniting and Strengthening America by Providing Appropriate Tools Required to Intercept and Obstruct Terrorism (USA PATRIOT ACT) Act of 2001, 2011.

[2] Bill Clinton. (1998, May) Presidential Decision Directive, NSC-63, Critical Infrastructure Protection. Internet.

[3] (1998, Nov) The Department of Defense Critical Infrastructure Protection Plan. Internet.

[4] CCSDS, "CCSDS 232.0-B-2 Telecommand (TC) Space Data Link Protocol," The Consultative Committe for Space Data Systems, Recommendation for Space Data Systems Standards 2003.

[5] CCSDS, "CCSDS 350.0-G-2 The Application of CCSDS Protocols to Secure Systems," The Consultative Committe for Space Data Systems, Informational Report 2006.

[6] CCSDS, "CCSDS 350.1-G-1 Security Threats Against Space Missions," The Consultative Committe for Space Data Systems, Report Concerning Space Data Systems Standards 2006.

[7] CCSDS, "CCSDS 200.0-G-6 Telecommand Summary of Concept and Rationale," 1987.

[8] (2010, January) Ham Test Online. [Online]. http://www.hamradiolicenseexam.com/question_pools/extra_2008/E1D.htm

[9] Howard Lipson, Evolutionary Design of Secure Systems – The First Step Is Recognizing the Need for Change, 2011.

[10] SSI, Space Security Executive Summary 2010, 2011.

[11] D. K. Sachdev, *Success stories in satellite systems*, D. K. Sachdev, Ed. Reston, VA: AIAA, 2009.

[12] Robert F. Dacey, "Commerical Satellite Security Should Be More Fully Addressed," United States General Account Office, Report to the Ranking Minority Member, Permanent Subcommittee on Investigations, Committee on Governmental Affairs, U.S. Senate 2002.

[13] George W. Bush. (2003, December) Homeland Security Presidential Directive 7: Critical Infrastructure Identification, Prioritization, and Protection. Internet.

[14] George W. Bush, National Security Presidential Directive, NSPD-49, U.S. National Space Policy, 2011.

[15] Bruno Pattan, *Satellite systems : principles and technologies*. New York: Van Nostrand Reinhold, 1993.

[16] Bruce R Elbert, *Introduction to satellite communication*. Boston: Artech House, 2008.

[17] Mark R Chartrand, *Satellite Communications for the Nonspecialist*.: SPIE - The International Society for Optical Engineering, 2004.

[18] Shawana P. Johnson. (2011, January) earthzine. [Online]. http://www.earthzine.org/2010/07/23/geospatial-applications-in-agriculture-and-global-food-security-an-nga-and-usda-project-success/

[19] USAF. (2011, January) Los Angeles Air Force Base. [Online]. http://www.losangeles.af.mil/library/factsheets/factsheet.asp?id=5514

[20] NOAA. (2011, January) National Oceanic and Atmospheric Administration. [Online]. http://www.noaa.gov/satellites.html

[21] Chad Schweitzer. (2011, January) design concepts. [Online]. http://www.design-concepts.com/blog/celebrate-navigation-and-timing-diversity

[22] ESA. (2011, January) European Objectives and Interests in Space Exploration. [Online]. http://esamultimedia.esa.int/docs/exploration/EuropeanThemes/European_Objectives_in_Space_Exploration.pdf

[23] NASA. (2011, January) NASA Solutions. [Online]. www.nasasolutions.com/at_home.html

[24] Zhili Sun, *Satellite Networking : principles and protocols.*: John Wiley \& Sons Ltd, 2005.

[25] Radiation Effects and Analysis Group. (2011, January) Radiation Effects and Analysis. [Online]. http://radhome.gsfc.nasa.gov/radhome/see.htm

[26] (2011, January) Goddard Space Flight Center. [Online]. http://heasarc.gsfc.nasa.gov/docs/rosat/gallery/misc_saad.html

[27] Clyde Space. (2011, January) Clyde Space EPS. [Online]. http://www.clyde-space.com/cubesat_shop/eps

[28] W.A. Dos Santos, A.M. da Cunha, and O.A. Martins, "Exploring round-trip engineering capabilities for satellite flight software projects," , vol. 2, 2005, p. 11 pp. Vol. 2.

[29] Brian Davis. (2011, January) Workshops on Spacecraft Flight Software. [Online]. http://flightsoftware.jhuapl.edu/files/FSW08_Davis.ppt

[30] CCSDS, "CCSDS 350.2-G-1 Encryption Algorithm Trade Survey," The Consultative Committe for Space Data Systems, Informational Report 2008.

[31] "Colony-I Space to Ground Interface Control Document, Core Elements [DRAFT], Volume 1 Command and Telemetry Definitions," 2010.

[32] Matt Bishop, *Computer Security*. New Jersey, USA: Addison-Wesley, 2003.

[33] Weiliang W. Zhao and Vijay Varadharajan, "An Approach to Unified Trust Management Framework," in *Collaborative computer security and trust management.*, 2009, ch. An Approach to Unified Trust Management Framework, pp. 111-134.

[34] Vijay Varadharajan, "Authorization and Trust Enhanced Security for Distributed Applications," in *Information Systems Security*, Sushil Jajodia and Chandan Mazumdar, Eds.: Springer Berlin / Heidelberg, 2005, vol. 3803, pp. 1-20, 10.1007/11593980_1.

[35] Weiliang Zhao, Vijay Varadharajan, and George Bryan, "A Unified Framework for Trust Management," , 2006, pp. 1-8.

[36] Weiliang Zhao, Vijay Varadharajan, and George Bryan, "Modelling Trust Relationships in Distributed Environments," , 2004, pp. 40-49.

[37] Weiliang Zhao, Vijay Varadharajan, and George Bryan, "General methodology for analysis and modeling of trust relationships in distributed computing," *Journal of Computers*, vol. 1, no. 2, pp. 42-53, 2006.

[38] Weiliang Zhao, Vijay Varadharajan, and George Bryan, "Analysis and Modelling of Trust in Distributed Information Systems," in *Information Systems Security*, Sushil Jajodia and Chandan Mazumdar, Eds.: Springer Berlin / Heidelberg, 2005, vol. 3803, pp. 106-119, 10.1007/11593980_8.

[39] Jose E. Fadul, Kenneth M. Hopkinson, Christopher A. Sheffield, James T. Moore, and Todd R. Andel, "Trust Management and Security in the Future Communication-Based "Smart" Electric Power Grid," in *44th Hawaii International Conference on System Sciences (HICSS)* , 2011, pp. 1-10.

[40] Kenneth P. Birman, *Reliable Distributed Systems*.: Springer Science + Business Media, 2005.

[41] Matt Blaze, Joan Feigenbaum, Joan Ioannidis, and Angelos Keromytis, "The KeyNote Trust Management System," IETF, RFC 1999.

[42] Matt Blaze, Using the KeyNote Trust Management System, 2001.

[43] Bin Yu and Munindar P. Singh, "A Social Mechanism of Reputation Management in Electronic Communities," in *In Proceedings of Fourth International Workshop on Cooperative Information Agents*, 2000, pp. 154-165.

[44] Matt Blaze, John Ioannidis, and Angelos Keromytis, "Experience with the KeyNote Trust Management System: Applications and Future Directions," in *Trust Management*, Paddy Nixon and Sotirios Terzis, Eds.: Springer Berlin / Heidelberg, 2003, vol. 2692, pp. 1071-1071, 10.1007/3-540-44875-6_21.

[45] Amirali Salehi-Abari and Tony White, "Towards con-resistant trust models for distributed agent systems," in *Proceedings of the 21st international jont conference on Artifical intelligence*, 2009, pp. 272-277.

[46] J. McDermott and C. Fox, "Using abuse case models for security requirements analysis," , 1999, pp. 55-64.

[47] Lei Zhang, Chengjin An, Quan Zhang, and Chaojing Tang, "Misbehavior Detection Algorithm in CCSDS Space Telecommand System," *Communications Letters, IEEE*, vol. 14, no. 8, pp. 746-748, 2010,

[48] Dan Balderston, "Space Segment Cyber Defense," Aerospace Corporation, El Segundo, Technical Presentation 2010.

[49] Silicon Labs, "C8051F12x-13x," Silicon Labs, Austin, Technical Report 2005.

[50] National Institute of Standards and Technology, "FIPS 197 ADVANCED ENCRYPTION STANDARD," NIST, FIPS Pub 197, 2001.

[51] niyazpk. (2011, January) Hoozi Resources. [Online]. http://www.hoozi.com/post/0m3lb/advanced-encryption-standard-aes-implementation-in-c-c

[52] Greg Hoglund and Gary McGraw, *Exploiting Software: How To Break Code.*: Addison-Wesley, 2008,

[53] William J. Lynn. (2010, January) DoDD 3020.40 : DoD Policy and Responsibilities for Critical Infrastructure. Internet.

[54] Bin Yu and Munindar P. Singh, "Detecting deception in reputation management," , 2003, pp. 73-80.

[55] Bin Yu and Munindar P. Singh, "An evidential model of distributed reputation management," , 2002, pp. 294-301.

[56] S. S. Yau, H. Davulcu, S. Mukhopadhyay, D. Huang, and Y. Yao, "Adaptable situation-aware secure service-based (AS3) systems," , 2005.

[57] R. Yahalom, B. Klein, and T. Beth, "Trust relationships in secure systems-a distributed authentication perspective," , 1993, pp. 150-164.

[58] J. Whittle, D. Wijesekera, and M. Hartong, "Executable misuse cases for modeling security concerns," , 2008, pp. 121-130.

[59] Maarten Van Steen, *Distributed Systems: Principles and Paradigms*, 2nd ed.: Prentice-Hall, 2008.

[60] Guttorm Sindre and Andreas L. Opdahl, "Eliciting security requirements with misuse cases," *Requirements Engineering*, vol. 10, pp. 34-44, 2005, 10.1007/s00766-004-0194-4.

[61] United States Senate, CYBER ATTACKS: THE NATIONAL PROTECTION PLAN AND ITS PRIVACY IMPLICATIONS, Hearings Before Congress, February 1, 2000, 2011.

[62] Sini Ruohomaa and Lea Kutvonen, "Trust Management Survey," in *Trust Management*, Peter Herrmann, Valérie Issarny, and Simon Shiu, Eds.: Springer Berlin / Heidelberg, 2005, vol. 3477, pp. 77-92, 10.1007/11429760_6.

[63] Robin Ruefle, The Role of Computer Security Incident Response Teams in the Software Development Life Cycle, 2011.

[64] Barbara Pusey, Dr. Carleen Maitland, Dr. Andrea Tapia, Dr. John Yen, and Barbara Pusey, A Survey of Trust Models in Agent Applications, Looks like a draft paper.

[65] Huaizhi Li and Mukesh Singhal, "Trust Management in Distributed Systems," *Computer*, vol. 40, no. 2, pp. 45-53, 2007.

[66] Soo-Yeon Kang, Sang-Kon Lee, and Koon-Ho Yang, "The COMS telecommand processing in the flight software," , 2009, pp. 1-3.

[67] Audun Jøsang, "The Right Type of Trust for Distributed Systems," , 1996, p. 119–131.

[68] C.B. Haley, R. Laney, J.D. Moffett, and B. Nuseibeh, "Security Requirements Engineering: A Framework for Representation and Analysis," *Software Engineering, IEEE Transactions on*, vol. 34, no. 1, pp. 133-153, 2008.

[69] Tyrone Grandison and Morris Sloman, "A Survey of Trust in Internet Applications," *IEEE Communications Surveys and Tutorials*, vol. 3, no. 4, 2000, http://www.comsoc.org/livepubs/surveys/public/2000/dec/index.html.

[70] T W Grandison, "Conceptions of trust: Definitions, constructs and models," , pp. 1-28.

[71] Mary Foote, Improving Defense in Depth for NASA's Mission Network, 2011, posted on October 31, 2003.

[72] Robert J. Ellison, Trustworthy Composition: The System Is Not Always the Sum of Its Parts, 2011.

[73] B. Dragovic, E. Kotsovinos, S. Hand, and P.R. Pietzuch, "XenoTrust: event-based distributed trust management," , 2003, pp. 410-414.

[74] Ioanna Dionysiou, "Dynamic and Composable Trust for Indirect Interactions," Doctoral Dissertation 2006.

[75] Ing-Ray Chen, Fenye Bao, Moonjeong Chang, and Jin-Hee Cho, "Trust Management for Encounter-Based Routing in Delay Tolerant Networks," , 2010, pp. 1-6.

[76] David W. Chadwick and Alexander Otenko, "The PERMIS X.509 role based privilege management infrastructure," *Future Generation Computer Systems*, vol. 19, no. 2, pp. 277-289, 2003.

[77] CCSDS, "CCSDS 350.3-G-1 Authentication/Integrity Algorithm Issues Survey," The Consultative Committe for Space Data Systems, Informational Report 2008.

[78] Matt Blaze, Joan Feigenbaum, John Ioannidis, and Angelos Keromytis, "The Role of Trust Management in Distributed Systems Security," in *Secure Internet Programming*, Jan Vitek and Christian Jensen, Eds.: Springer Berlin / Heidelberg, 1999, vol. 1603, pp. 185-210, 10.1007/3-540-48749-2_8.

[79] Matt Blaze, Joan Feigenbaum, and Jack Lacy, "Decentralized Trust Management," 1996.

[80] F. Azzedin and M. Maheswaran, "Evolving and managing trust in grid computing systems," , vol. 3, 2002, pp. 1424 - 1429 vol.3.

[81] Robert K. Ackerman, Space Now a Contested Venue, 2009.

[82] S. Aboulwafa and R. Bahgat, "DiReCT: Dirichlet-based Reputation and Credential Trust management," , 2010, pp. 1-8.

[83] Conversation With Brian Davis SGSS, 2011.

[84] Donovan Artz and Yolanda Gil, "A survey of trust in computer science and the Semantic Web," *Web Semantics*, vol. 5, no. 2, pp. 58-71, 2007.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 074-0188*

1. REPORT DATE *(DD-MM-YYYY)* 24-03-2011	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From – To)* Mar 2010 – Mar 2011

4. TITLE AND SUBTITLE Trust Management and Security in Satellite Telecommand Processing	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Duncan, Mark C., Capt, USAF	**5d. PROJECT NUMBER** 11G222
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology, Graduate School of Engineering and Management (AFIT/EN), 2950 Hobson Way, WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GCO/ENG/11-03
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Office of Scientific Research, Attn: Dr. Robert Bonneau, 875 N. Randolph St, Ste 325 Rm 3112, Arlington, VA 22203, DSN: 426-9545, Email: robert.bonneau@afosr.af.mil	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFOSR/NL
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT
New standards and initiatives in satellite system architecture are moving the space industry to more open and efficient mission operations. Primarily, these standards allow multiple missions to share standard ground and space based resources to reduce mission development and sustainment costs. With the benefits of these new concepts comes added risk associated with threats to the security of our critical space assets in a contested space and cyberspace domain. As one method to mitigate threats to space missions, this research develops, implements, and tests the Consolidated Trust Management System (CTMS) for satellite flight software.
The CTMS architecture was developed using design requirements and features of Trust Management Systems (TMS) presented in the field of distributed information systems. This research advances the state of the art with the CTMS by refining and consolidating existing TMS theory and applying it to satellite systems. The feasibility and performance of this new CTMS architecture is demonstrated with a realistic implementation in satellite flight software and testing in an emulated satellite system environment. The system is tested with known threat modeling techniques and a specific forgery attack abuse case of satellite telecommanding functions. The CTMS test results show the promise of this technique to enhance security in satellite flight software telecommand processing. With this work, a new class of satellite protection mechanisms is established, which addresses the complex security issues facing satellite operations today. This work also fills a critical shortfall in validated security mechanisms for implementation in both public and private sector satellite systems.

15. SUBJECT TERMS
Trust Management, Satellite, Flight Software, Security

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Kenneth M. Hopkinson (ENG)
REPORT U	**ABSTRACT** U	**c. THIS PAGE** U	U	123	**19b. TELEPHONE NUMBER** *(Include area code)* *(937)255-3636 x4579, kenneth.hopkinson@afit.edu*