CORE FLIGHT SYSTEM
CRYPTOGRAPHY LIBRARY

John Patrick Lucas

Thesis submitted to the Benjamin M. Statler College of
Engineering and Mineral Resources at
West Virginia University in partial fulfillment of the
requirements for the degree of

Master of Science
in
Electrical Engineering

Dr. Powsiri Klinkhachorn, Ph.D., LCSEE, Committee Chairperson
Dr. Roy Nutter, Ph.D., LCSEE
Dr. Afzel Noore, Ph.D., LCSEE

Lane Department of Computer Science and Electrical Engineering
Morgantown, West Virginia
2017

Keywords: CCSDS, cFS, Core Flight System, Cryptography,

Copyright 2017 John Patrick Lucas
ABSTRACT

CORE FLIGHT SYSTEM
CRYPTOGRAPHY LIBRARY

John Patrick Lucas

The NASA Independent Verification and Validation (IV&V) Program has been tasked with performing testing for the Consultative Committee for Space Data Systems (CCSDS) Space Data Link Security (SDLS) Extended Procedure (EP) protocol. This testing has allowed for both the expansion of inter-agency collaboration and the publishing of the SDLS-EP Standard. This protocol is a “data processing method for space missions that need to apply authentication and/or confidentiality to the contents of Transfer Frames used by the Space Data Link Protocols over a space link” (CCSDS)\(^1\). The interest for security on satellites stems from threats including but not limited to data corruption, interception, access control, and physical attacks.

Protocol testing has been performed in the cloud between the European Space Agency and NASA. The flight software used on-board the simulated spacecraft was the Core Flight System (cFS). A cFS cryptography library, named Crypto, was developed to implement the SDLS-EP procedures and allow for easy integration into existing command ingest and telemetry output applications via a single security function. This library allows the spacecraft to communicate using the standardized Telecommand and Telemetry protocols while still publishing received CCSDS messages to the cFS software bus for application processing. The cFS Crypto library is currently undergoing the NASA release process to become open-source software. The sole designer, developer, and tester is looking to expand and increase current development speed by leveraging the community.

\(^1\) [https://public.ccsds.org/Pubs/355x0b1.pdf](https://public.ccsds.org/Pubs/355x0b1.pdf)
ACKNOWLEDGEMENTS

This thesis would not be possible without the support, guidance, and encouragement of those around me. Firstly, I would like to expressly mention and thank my committee chair, Dr. Klinkhachorn, for his support throughout both my undergraduate and graduate work. Special thanks as well to Dr. Nutter and Dr. Noore for serving as both my teachers and committee members.

NASA IV&V, and in particular the JSTAR and ITC groups, provided the foundation for this work and will continue to provide oversight in the efforts of expansion. In particular, I would like to name Justin Morris, Brandon Bailey, Scott Zemerick, Dustin Geletko, Jason Battin, Matthew Grubb, Justin McCarty, and Mark Pitts, whom have all contributed immensely to this effort.

Finally, I would like to thank my family and Amrita Kaur Anand for standing by my side throughout my years of graduate school, not only dealing with my limited time, but my limited focus and attention due to my work.

_Nemo vir est qui mundum non reddat meliorem_
TABLE OF CONTENTS

ABSTRACT .......................................................................................................................... ii
ACKNOWLEDGEMENTS .................................................................................................... iii
TABLE OF CONTENTS ....................................................................................................... iv
LIST OF FIGURES ................................................................................................................ vi
LIST OF TABLES .................................................................................................................. vii
LIST OF SYMBOLS / NOMENCLATURE ........................................................................... ix
Chapter 1 | INTRODUCTION ............................................................................................... 1
  1.1. Statement of the Problem ....................................................................................... 1
  1.2. Motivation ................................................................................................................ 2
  1.3. Thesis Organization ............................................................................................... 3
Chapter 2 | LITERATURE REVIEW ...................................................................................... 4
  2.1. Cryptographic Principles ....................................................................................... 4
  2.2. Threat and Risk ...................................................................................................... 6
  2.3. Defense in Depth .................................................................................................... 9
  2.4. Espionage in Action ............................................................................................. 10
  2.5. The Consultative Committee for Space Data Systems ..................................... 13
  2.6. Galois Counter Mode AES ................................................................................ 29
  2.7. Space Data Link Security – Extended Procedures ........................................... 30
Chapter 3 | DESIGN ............................................................................................................. 39
  3.1. Starting Point .......................................................................................................... 39
  3.2. High Level Design ................................................................................................ 43
  3.3. Software Details .................................................................................................... 47
  3.4. Architecture ........................................................................................................... 50
  3.5. Deployment ............................................................................................................ 51
  3.6. Operations ............................................................................................................. 53
Chapter 4 | TESTING ............................................................................................................ 57
  4.1. Local Testing .......................................................................................................... 57
4.2. Interoperability Testing ................................................................. 59
4.3. Simulation To Flight 1 – Example Mission Implementation ...... 63

Chapter 5 | CONCLUSION ........................................................................... 65

  5.1. Summary .......................................................................................... 65
  5.2. Future Work .................................................................................... 65
  5.3. Remaining Questions ...................................................................... 67

Chapter 6 | BIBLIOGRAPHY ....................................................................... 69

Chapter 7 | APPENDIX .................................................................................. 71

  5.4. Open Source Paperwork ................................................................. 71
  5.5. Presentation .................................................................................... 75
LIST OF FIGURES

Figure 1 - Cryptographic Hash Example .................................................................................. 5
Figure 2 - Message Authentication Code Diagram .................................................................. 6
Figure 3 - Generic Threats to Spacecraft [5] ........................................................................... 7
Figure 4 - Defense in Depth Model and Map of a Walled City [11] ........................................ 10
Figure 5 - Rad1o Badge [13] .................................................................................................. 11
Figure 6 – Turla APT Scheme [14] ......................................................................................... 12
Figure 7 - SLE State Transition Diagram ............................................................................... 14
Figure 8 - SLE Transaction Example ....................................................................................... 15
Figure 9 - Satellite Identification Relations [2] ...................................................................... 17
Figure 10 - Telecommand Transfer Frame [2] ....................................................................... 18
Figure 11 - Telecommand Primary Header [2] ....................................................................... 19
Figure 12 - Frame Error Control Field Calculation [2] ............................................................ 20
Figure 13 - Telecommand Security Frame [2] ....................................................................... 21
Figure 14 - Telemetry Transfer Frame [16] .......................................................................... 22
Figure 15 – Telemetry Primary Header [16] .......................................................................... 23
Figure 16 - Telemetry Data Field [16] ................................................................................. 24
Figure 17 - TM Frame CLCW [2] .......................................................................................... 25
Figure 18 - Transfer Frame with SDLS [6] ............................................................................ 26
Figure 19 - Security Header [6] ............................................................................................. 26
Figure 20 - Security Trailer [6] ............................................................................................ 27
Figure 21 – CMAC Generation [17] ....................................................................................... 27
Figure 22 - Galois Counter Mode AES Diagram [17] ............................................................. 29
Figure 23 – Cryptographic Key Lifecycle [4] .......................................................................... 30
Figure 24 - Variable State Model for Security Association Management [4] ......................... 31
Figure 25 - Extended Procedures PDU [4] ............................................................................. 32
Figure 26 - Frame Security Report [4] ................................................................................... 34
Figure 27 - Space Packet Frame [18] ..................................................................................... 35
Figure 28 - Space Packet Primary Header [18] ...................................................................... 36
Figure 29 - Space Packet Secondary Header [18] ................................................................. 37
Figure 30 - Packet Data Field Diagram [19] ................................................................. 38
Figure 31 - Previous Work Architecture Diagram [20] ......................................................... 40
Figure 32 - Previous Work Data Flow [20] ........................................................................... 41
Figure 33 - AES Unit Test .................................................................................................... 42
Figure 34 - GCM Unit Test .................................................................................................. 42
Figure 35 - Previous Work Interoperability Testing [20] ....................................................... 43
Figure 36 – ESA SDLS-EP Testing Architecture [21] ........................................................... 44
Figure 37 - Security Unit and Proxy [21] ............................................................................. 45
Figure 38 - Interoperability Testing using a VLAN [22] ...................................................... 46
Figure 39 - SLE_IVV Software Diagram ............................................................................. 49
Figure 40 - User Commands Script .................................................................................... 50
Figure 41 - Cryptography Library Overview ....................................................................... 51
Figure 42 - Cryptography Library Local Testing ................................................................. 58
Figure 43 – Cryptography Library Demo Terminal 1 ............................................................. 61
Figure 44 – Cryptography Library Demo Terminal 2 ............................................................. 62
Figure 45 – Cryptography Library Demo Terminal 3 ............................................................. 62
Figure 46 – Cryptography Library Demo Terminal 4 ............................................................. 62
Figure 47 – Cryptography Library Demo VNC View ............................................................. 63
Figure 48 - Simulation to Flight 1 Mission Patch ................................................................. 64
Figure 49 – NOS³ Logo .................................................................................................... 67
LIST OF TABLES

Table 1 – Types of Threats to Spacecraft .................................................................................. 8
Table 2 - Vulnerabilities of Spacecraft [10] ................................................................................. 9
Table 3 - EP Key Service Identifications ......................................................................................... 32
Table 4 – EP SA Service Identifications ......................................................................................... 33
Table 5 – EP MC Service Identifications ......................................................................................... 33
Table 6 - Mission Specific Options Defined ....................................................................................... 54
Table 7 - Debug Fields ....................................................................................................................... 56
### LIST OF SYMBOLS / NOMENCLATURE

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AOS</td>
<td>Advanced Orbiting Systems</td>
</tr>
<tr>
<td>API</td>
<td>Application Program Interface</td>
</tr>
<tr>
<td>APID</td>
<td>CCSDS Application ID</td>
</tr>
<tr>
<td>CCSDS</td>
<td>The Consultative Committee for Space Data Systems</td>
</tr>
<tr>
<td>CFS</td>
<td>Core Flight System</td>
</tr>
<tr>
<td>CNES</td>
<td>Centre National d’Etudes Spatiales</td>
</tr>
<tr>
<td>COP-1</td>
<td>Communications Operation Procedure - 1</td>
</tr>
<tr>
<td>EP</td>
<td>Extended Procedures</td>
</tr>
<tr>
<td>FCLTU</td>
<td>Forward Communications Link Transmission Unit</td>
</tr>
<tr>
<td>FECEF</td>
<td>Forward Error Control Field</td>
</tr>
<tr>
<td>GCM AES</td>
<td>Galois Counter Mode Advanced Encryption Standard</td>
</tr>
<tr>
<td>GVCID</td>
<td>Global Virtual Channel ID</td>
</tr>
<tr>
<td>IV</td>
<td>Initialization Vector</td>
</tr>
<tr>
<td>LDP</td>
<td>Logical Data Path</td>
</tr>
<tr>
<td>MAC</td>
<td>Message Authentication Code</td>
</tr>
<tr>
<td>OCF</td>
<td>Operational Control Field</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>OTAR</td>
<td>Over The Air Re-keying</td>
</tr>
<tr>
<td>PDU</td>
<td>Payload Data Unit</td>
</tr>
<tr>
<td>RCF</td>
<td>Return Channel Frames</td>
</tr>
<tr>
<td>SCID</td>
<td>Space Craft ID</td>
</tr>
<tr>
<td>SDLP</td>
<td>Space Data Link Protocol</td>
</tr>
<tr>
<td>SDLS</td>
<td>Space Data Link Services</td>
</tr>
<tr>
<td>SLE</td>
<td>Space Link Extension</td>
</tr>
<tr>
<td>SN</td>
<td>Sequence Number</td>
</tr>
<tr>
<td>SPI</td>
<td>Security Parameter Index</td>
</tr>
<tr>
<td>TC</td>
<td>Tele-Command</td>
</tr>
<tr>
<td>TM</td>
<td>Telemetry</td>
</tr>
<tr>
<td>-----</td>
<td>-------------</td>
</tr>
<tr>
<td>VCID</td>
<td>Virtual Channel ID</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Cross Support</td>
<td>“Cross Support is when multiple organization provide space mission</td>
</tr>
<tr>
<td></td>
<td>functionality.” [1]</td>
</tr>
<tr>
<td>Delimited</td>
<td>“Having a known (and finite) length; applies to data in the context of</td>
</tr>
<tr>
<td></td>
<td>data handling.” [2]</td>
</tr>
<tr>
<td>Forwarding</td>
<td>“The act of transferring data from its source towards its destination,</td>
</tr>
<tr>
<td></td>
<td>which may be in space or on the ground.” [3]</td>
</tr>
<tr>
<td>Initiator</td>
<td>“The Initiator of an SDLS Expended Procedure is one of the two peers</td>
</tr>
<tr>
<td></td>
<td>involved in an SDLS communication session. The Initiator is managing the</td>
</tr>
<tr>
<td></td>
<td>SDLS session parameters, provides the necessary resources to execute a</td>
</tr>
<tr>
<td></td>
<td>procedure, and always initiates a procedure.” [4]</td>
</tr>
<tr>
<td>Mission Phase</td>
<td>“A period of a mission during which specific communications characteristics</td>
</tr>
<tr>
<td></td>
<td>are fixed. The transition between two consecutive Mission Phases may cause</td>
</tr>
<tr>
<td></td>
<td>an interruption of the communications services.” [2]</td>
</tr>
<tr>
<td>Octet</td>
<td>“An 8-bit word.” [3]</td>
</tr>
<tr>
<td>Physical Channel</td>
<td>“A stream of bits transferred over a space link in a single direction.”</td>
</tr>
<tr>
<td></td>
<td>[3]</td>
</tr>
<tr>
<td>Recipient</td>
<td>“The Recipient of an SDLS Extended Procedure is one of the two peers involved in an SDLS communication session. It configures the SDLS session parameters based on procedures and instruction initiated by the Initiator.” [4]</td>
</tr>
<tr>
<td>Risk</td>
<td>“Possibility that a particular threat will adversely impact an information</td>
</tr>
<tr>
<td></td>
<td>system by exploiting a particular vulnerability.” [5]</td>
</tr>
<tr>
<td>Routing</td>
<td>“The process of selecting paths from origins to destinations in a network.”</td>
</tr>
<tr>
<td></td>
<td>[3]</td>
</tr>
<tr>
<td>Space Communications Protocol</td>
<td>“A communications protocol designed to be used over a space link, or a network that contains one or multiple space links.” [3]</td>
</tr>
<tr>
<td>Space Link</td>
<td>“A communications link between a spacecraft and its associated ground system or between two spacecraft. A space link consists of one or more Physical Channels in one or both directions.” [3]</td>
</tr>
<tr>
<td>Threat</td>
<td>“A potential cause of an incident that may result in harm of systems and organization” [5]</td>
</tr>
</tbody>
</table>
CHAPTER 1 | INTRODUCTION

1.1. Statement of the Problem

Cybersecurity incidents have made headlines in recent years as criminals manipulate vulnerabilities in systems for illicit purposes, financial gain, and fanfare. Satellites perform a vast number of operations, ranging from military use to scientific endeavors, and are prime candidates for attack. As space becomes commercialized and crowded, the need for an accepted standard that provides security features and protects against various forms of attack at a low cost is growing.

The Consultative Committee for Space Data Systems (CCSDS) is an international standards group, led by civil space agencies, for the development of communications and data standards for spaceflight with the goal of enhancing governmental and commercial interoperability and cross-support, while also reducing risk, development time, and project cost. The NASA Independent Verification and Validation (IV&V) facility has collaborated with CCSDS on previous work involving the Space Data Link Security (SDLS) Core Protocol (CP). This protocol provided a security header and trailer along with associated procedures for applying data authentication and/or confidentiality at the Data Link Layer. [6] SDLS Testing was performed previously, at reduced expense to the agency, at NASA IV&V by the Independent Test Capability (ITC) team, a sub-set of the Jon McBride Software Testing and Research (JSTAR) laboratory. The initial effort was based upon the Global Precipitation Measurement (GPM) mission and implemented the Advanced Orbiting Systems (AOS) protocol in the return or telemetry space link. Experience with SDLS led to providing interoperability testing of the SDLS - Extended Procedures (EP) protocol in baseline mode. This also allowed for both the expansion of inter-agency communication, the publishing of the SDLS-EP Standard, the publishing of an inter-agency cloud-testing standard, and the creation of the Cryptography Library, the essence of this thesis.

The Cryptography Library has been designed to provide a generic starting point for the introduction of security on space missions. The first phase of this thesis introduces the cryptographic principles,
recent security events, and protocols, while the second provides further background into previous work and the overall design of the package. Testing is then covered at both the intra or local and interoperability levels before concluding.

This library can easily be utilized in other frameworks as well and, at a minimum, provides a tested baseline for further agency and user development to occur. The library has been made available via the NASA Open Source Agreement to the public through NASA Case No. GSC-17,776-1.

1.2. Motivation

The interest in security on spacecraft systems stems from current issues on the ground that are driving the need for increased cyber security across all systems. These issues include hackers seeking profit by means of data interception, disabling the system, or even utilizing the system for ill purpose, such as an attack on another satellite or spacecraft that could potentially cause harm to one or more persons. Attacks such as these occur on a daily basis on the ground and have spread into space, as elaborated upon in Section 2.4.

Security on Space Agency spacecraft can benefit from current standards utilized by militaries and from yet unproven technologies. With the commercialization of space and the introduction of small spacecraft, the need for a broadly accepted and proven technology exists. In the short term, lightweight plug-and-play software based libraries or applications are a solution. As satellites become more standardized, the uniqueness and obscurity that has protected them in the past is removed, further driving the need for a long term solution.

A recent Chatham House paper on space and cybersecurity points to the increasingly blurred lines between ‘offensive’ and ‘defensive’ activities in cyber and space, given that, technologically, offence is easier and more cost-effective than defense. More advanced countries are increasingly vulnerable to attack from less developed states, and from terrorists groups and other such organized criminals. In common with other domains such as the civil nuclear industry, the lack of documented or reported events in the space cyber domain leads to a false sense of security: little seems to be happening, little is likely to happen, and so what is the point in adopting any countermeasures? But the evidence suggest that imagining the risk to be small could be a fatal blunder.

Secure encryption seems to be the most plausible response to cyber threats to space assets, although it has its limits. Some security is better than no security – as long as the experts know what the security is capable of providing and what its limitations are. Part of the problem appears to be that neither the cyber community nor the space community understand the security requirements and vulnerabilities of each other’s domain. However, the cyber and space communities do not just lack knowledge; they also need a
wider understanding of the concept of security. The biggest limit to security might be the high costs that the different stakeholders are faced with; not all of them are prepared to spend a considerable amount of money to protect their systems.

The transition observed is truly toward the commoditization of space, a trend away from military and research, to one where ‘the market’ (in which there is a persistent need to innovate in a data-hungry world) holds sway. The pace of change can only quicken as launch systems become cheaper (through ‘low-cost access to space’ initiatives) and more reliable. In this unique environment, the world is on the cusp of a new dynamic in which the issue of cybersecurity has not even caught up with the old, just as major change in market forces and corresponding supply chains is under way. What is becoming increasingly obvious is that there needs to be a radical review of cybersecurity in space. The universal access now provides just such an opportunity for significant changes.

There is an urgent requirement to develop a space cyber system regime that will inform and organize policy efforts and subordinate strategies, while remaining federally networked rather than controlled from a center or hierarchically driven. [7]

Traditionally, security mechanisms have not been employed on civilian space missions. In recognition of the increased threat, there has been a steady migration towards the integration of security services and mechanisms. For example, ground network infrastructures typically make use of controlled or protected networks. However, telecommands, telemetry, and science payload data, are still, for the most part, transmitted over unencrypted and unauthenticated radio frequency (RF) channels. As the threat environment becomes more hostile, this concept of operation becomes much more dangerous. [8]

1.3. Thesis Organization

This section concludes Chapter 1. Chapter 2 of this thesis is a literature review specifically detailing cryptographic principles, relevant current events, and protocol backgrounds. Chapter 3 describes the design of the Cryptography Library, beginning with previous work and expanding into deployment and suggested operations and procedures. Chapter 4 describes multiple methods of testing and presents results. In Chapter 5, conclusions, future work, and lessons learned are provided.
CHAPTER 2 | LITERATURE REVIEW

2.1. Cryptographic Principles

“Confidentiality is defined as the assurance that information is not disclosed to unauthorized entities or processes.” [9] As spacecraft require a radio frequency (RF) space link, the only means to achieve confidentiality is through encryption. Encryption allows only the initiator to transform ordinary data, plaintext, into unintelligible data, ciphertext, for safe transmission to the recipient. The sender and receiver share a secret key that makes the reverse of this process, decryption, possible. Encryption eliminates the threat of a third party interpreting the data.

Integrity of data can be summarized as accuracy. Is the data received the same as what was sent? Many threats can be eliminated with proper error checking and verification. The issue of integrity in space link communications has always been considered due to environmental factors and noise on RF links. One method of validating integrity is the use of a cryptographic hash function. A hash function is a one-way process that generates a fixed-size hash value for a variably sized input. This is non-reversible, unlike a cyclical redundancy check (CRC), and responds to any minor modifications with a different hash value. An example of this can be seen in Figure 1. It is infeasible that two messages with the same hash value will be generated.
“Authentication can be used to uniquely identify a person or entity.” [9] This allows access control to be implemented, as well as the logging of activities for traceability. “Authentication is implemented by either hash-based or cipher-based message authentication codes. It can also be implemented by use of digital signatures.” [9] A Message Authentication Code (MAC) is utilized in the Space Data Link Security (SDLS) Protocol discussed in Section 2.5.4. This MAC is similar to a hash, except a secret key is needed to generate the MAC. This MAC also guarantees that the message has not been altered. A depiction of the entire process, sender to receiver, is provided in Figure 2.
Availability is the most difficult cryptographic principle to obtain on a spacecraft. In software, all that may be done is ensuring that messages are received in sequential order and that requests for retransmission are performed. Additionally, bounds for certain values must be checked quickly, as that packet may be ignored so that others can be processed. Examples of what drives this need are captured in the next section.

2.2. Threat and Risk

Threat and risk are defined before the types of attacks currently known to the space industry are discussed, along with the risk this places upon the spacecraft and the owners. A deep dive into CCSDS protocols in use for this work follows in Section 2.5. These protocols are layered and broken down from the top level implemented and address the issues captured in this section.

“A ‘threat’ is a function of a threat agent’s capability and intent to do harm whereas ‘risk’ is a function of the probability that an organization will be targeted and the harm that might be caused. To further distinguish the difference:

- Threat = Capability x Intent;
• Risk = Probability x Harm.” [5]

With the commercialization of space and the growing number of satellites in orbit, the number of threats and amount of risk are increasing as well. As new capabilities in space become available, the risk increases and further drives interest in security. A large number of spacecraft, including large missions, do not have any type of security, further emphasizing the need for this protocol and additional work in the realm of security in the space industry. “In the past, space missions using CCSDS Recommended Standards were typically thought of as ‘civil’ and ‘scientific’ missions that were not likely targets of malicious attackers, unlike military missions that would be targeted and have traditionally been highly protected.” [5] A number of these recent attacks are detailed in Section 2.4, but understanding the threats is vital in determining the amount of risk.

“A threat agent (or threat source) can be human or non-human and can be intentional or unintentional.” [5] This section will detail potential types of threats to space systems. These threats originate from multiple sources. Each component or piece used on a mission has the potential to contain threats against it specifically. Some of these are captured in Figure 3 and Table 1 below.

Figure 3 - Generic Threats to Spacecraft [5]
Corruption Data corruption can be linked to an attacker, but is also a standard problem with space links or communications. The environmental noise, both in space and on Earth, causes a host of problems from radiation Single Event Upsets (SEUs), to requiring large amounts of power at the transmit location to reach the destination or to sample at the receiving end.

Interception An attacker can capture unprotected data. The result of this can range from non-critical, in the case of a large portion of NASA transmissions where all of the data is made public anyways, to critical, in the case of government or military satellites carrying extremely sensitive data.

Masquerade An attacker can ‘masquerade’ or pose as either the ground station, Mission Operations Center (MOC), or the spacecraft. This could enable the communication of sensitive data to an attacker or, in the worst case, enable the attacker to obtain control of the spacecraft or target.

Replay The mission of a replay attack is to broadcast previously valid data in an attempt to disrupt operations in some way. Attackers can also reply modified messages received via interception in an attempt to gain unauthorized access.

Denial of Service Denial of service attacks can occur in a multitude of ways, including jamming, disrupting the communications paths, or consuming all available resources. This type of attack, although seemingly harmless on the surface, can lead to critical operations failure.

Table 1 – Types of Threats to Spacecraft
“Invincibility is a matter of defense; vulnerability is a matter of attack.” – Sun Tzu

“Civilian space mission(s) are supported by a large system of interconnected resourced and assets both in space and on the ground (e.g., computer systems, communication devices, processors, etc.). Each of these entities potentially has vulnerabilities that could be exploited by a threat agent. A successful attack may result in mission impact. Mission impacts can range from insignificant (such as a software crash) to catastrophic (loss of mission).” [5]

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backdoors</td>
<td>“Mechanisms used to access undocumented features or interfaces not intended for end users.”</td>
</tr>
<tr>
<td>Hardcoded Credentials</td>
<td>“Undocumented credentials that can be used to authenticate in documented interfaces expected to be available for user interaction.”</td>
</tr>
<tr>
<td>Insecure Protocols</td>
<td>“Documented protocols that pose a security risk.”</td>
</tr>
<tr>
<td>Undocumented Protocols</td>
<td>“Undocumented protocols, or protocols not intended for end users, which pose a security risk.”</td>
</tr>
<tr>
<td>Weak Password Reset</td>
<td>“Mechanism that allows resetting other’s passwords.”</td>
</tr>
</tbody>
</table>

Table 2 - Vulnerabilities of Spacecraft [10]

2.3. Defense in Depth

The principle of defense in depth is one commonly recommended by security professionals. One can visualize this by comparing cyber defense to a castle. The keep inside is the precious data or functionality you are safeguarding, i.e. the royalty. Around this keep are the various defenses arranged in a layered approach. To reach the royalty, you must first cross the mote, get through the wall and across the courtyard to reach the walls of the keep. This approach is ideal in the case that a single defense mechanism has a weakness that can be exploited, such as a key to the keep. This allows for another to chance at deterring an attack. The levels of information security are detailed in comparison to the keep analogy in Figure 4.
Most satellite mission operations centers practice standard security procedures throughout the data transmission process for the ground segment. Protocols to support interoperability at this level have been published and implemented internationally on various missions and facilities, such as the deep space network and James Webb Space Telescope (JWST). Due to the amount of data dissemination to scientists and their internet connection, standard cyber security practices are employed at these centers. Satellites themselves typically hide behind this fact, along with the argument of obscurity utilizing space links to avoid implementing security measures. Security is only as good as the easiest path, or weakest link, to the end goal of the attacker.

In 2011, a NASA audit reported that multiple servers were accessible for exploit via the internet. This audit was performed to re-check the status after another troubled audit the previous year. The report also references previous attacks in 2009 where machines were compromised and connected to various networks, along with the theft of 22GB of data from a JPL IT team. [12]

2.4. Espionage in Action

The following section provides summaries of satellite hacking events that have occurred within the past 10 years:
2.4.1. Chaos Communication Camp

At Chaos Communication Camp, in 2015, organizers distributed radio badges, shown in Figure 5. These devices are approximately the same size as a Raspberry Pi and include an antenna, allowing the user to intercept traffic on the Iridium satellite network. As is, they were able to intercept approximately 20% of the pager traffic on the network due to the cheap antenna that was used. With a proper antenna the interception rate could be increased to 100%. “The problem isn’t that Iridium has poor security. It’s that is has no security.” [13] After this camp, additional work has been done to intercept other types of traffic on the network utilizing the same device. Technologies like this have advanced so rapidly that in 2008 Iridium stated that “the complexity of the Iridium air interface makes the challenge of developing an Iridium L-Band monitoring device very difficult and probably beyond the reach of all but the most determined of adversaries.” Following the ending of the camp the organizer mentioned that “It’s kind of a myth that satellite hacking is hard.” [13]

![Figure 5 - Rad1o Badge](image)

2.4.2. SATCOM

IOActive in Seattle, Washington, performed an analysis of the security on Satellite Communications (SATCOM) and published a comprehensive study in 2014. This analysis utilized publically available firmware to search for vulnerabilities, and in many cases did not even require the physical device to find
vulnerabilities. Types of vulnerabilities found included poor protocols, backdoors, and hardcoded credentials allowing attackers to modify communications and even take control of the device. A single compromised device has the potential to expose other portions of the SATCOM infrastructure and affect all customers. “The findings of IOActive’s research should serve as an initial wake-up call for both the vendors and users of the current generation of SATCOM technology.” According to Santamarta, “encryption use isn’t as common as it should be. SATCOM services are expensive and data charges are a major drawback when introducing secure communications.” [10]

### 2.4.3. Turla Advanced Persistent Threat

“Turla is a sophisticated Russian cyber-espionage group, believed to be sponsored by the Russian government that has targeted a number of government, military, embassy, research, and pharmaceutical organizations in more than 45 countries including China, Vietnam, and the United States.” [14] These hackers have been able to exploit older satellites that do not support encrypted connections. Due to the large broadcast area, they can operate anywhere within it to effectively mask the true location. Once a satellite is in range, the attackers can monitor the traffic and attack other users of the same service. A diagram of this process is shown below in Figure 6.

![Figure 6 – Turla APT Scheme](image-url)
2.4.4. NASA / NOAA

To give some examples of the vulnerability of satellite systems, a draft report of the US Congress in 2011 recorded that at least two US environment-monitoring satellites has suffered interference four or more times in 2007 and 2008. A Landsat-7 Earth observation satellite built by NASA and managed by the US Geological Survey experienced 12 or more minutes of interference in October 2007 and July 2008. A NASA-managed Terra AM-1 Earth observation satellite suffered similar interference for two minutes or more on a single day in June 2008, and at least nine minutes on one day in October 2008. The US National Oceanographic and Atmospheric Administration (NOAA) reported that its Satellite Data Information System was taken offline in September 2014 after a serious hacking incident; this denied high volumes of data to weather forecasting agencies around the world for 48 hours. [7]

2.5. The Consultative Committee for Space Data Systems

This section starts at the highest level of CCSDS protocol used in this work, the Space Link Extension (SLE) service, and breaks down each into the fields and additional protocols underneath. Each service encapsulates those of the lower levels down to the Space Data Link Security Extended Procedures frame through the use of the payload data unit or PDU.
2.5.1. Space Link Extension

The operating procedure for the space link extension services includes three basic states: unbound, ready, and active. The service user will bind to the service provider, allowing for the first state change from unbound to ready. Upon receipt, the service provider will reply with a bind return. The service provider will always provide a response to the service user throughout the process, designated as an ‘operation’-return. Once in the ready state, the start request can be sent from the service user allowing for transition into the active state. The active state allows TRANSFER-DATA functions to be used. A simplified state transition diagram is provided below in Figure 7 along with an example operational scenario and Figure 8.

![Figure 7 - SLE State Transition Diagram](image)
The procedure detailed in this section works for both the Forward Communications Link Transmission Unit (FCLTU) and Return Channel Frame (RCF), sometimes referred to as the telecommand and telemetry links. Each direction has a unique configuration that allows mission-specific functionality to be implemented. “The user is isolated from the details of the individual organization’s SLE System.” [1]

2.5.2. Telecommand

The TC Protocol is used over the space link from the ground station to the spacecraft. “The major functions performed by this protocol are (1) segmentation and blocking of service data units and (2) transmission control of service data units.” “The TC Space Data Link Protocol provides the capability to break large services data units into relatively small pieces (segmentation) and to group small service
data units into larger pieces (blocking).” [2] This allows for mission optimization of the space link and provides a means for automatic retransmission of data through the Communications Operation Procedure (COP) elaborated upon in the next section.

“The protocol entity performs the following protocol functions:

a) generation and process of protocol control information (i.e., headers and trailers) to perform data identification, loss detection, and error detection;
b) segmenting and blocking of service data units to transfer data units of various sizes in protocol data units suitable for efficient transfer;
c) multiplexing/demultiplexing in order for various service users to share a single Physical Channel;
d) retransmission of missing protocol data units, rejection of out-of-sequence and duplicated protocol data units, and control of sequence-control mechanisms at sending and receiving ends to guarantee complete and in-order delivery;
e) flow control.

If the protocol entity supports the optional SDLS protocol, then it uses the functions of SDLS to apply the configured security features.

The protocol entity does not perform the following protocol functions:

a) connection establishment and release;
b) management or configuration of the SDLS protocol.” [2]

2.5.2.1. Segmentation / Addressing

“A key feature of the TC Space Data Link Protocol is the concept of ‘Virtual Channels’. The Virtual Channel facility allows one Physical Channel to be shared among multiple higher-layer data streams, each of which may have different service requirements. A single Physical Channel may therefore be divided into several separate logical data channels, each known as a ‘Virtual Channel’ (VC). Each Transfer Frame transferred over a Physical Channel belongs to one of the Virtual Channels of the Physical Channel.” [2] These channels can be thought of as an address or identifier.

Three fields make up the entire Global Virtual Channel Identifier (GVCID). These include the Transfer Frame Version Number (TFVN), Spacecraft Identifier (SCID), and Virtual Channel Identifier (VCID). These can be combined to provide the GVCID as seen in Figure 9 below. A single spacecraft would have a unique GCVID while spacecraft utilizing a single space link could share an MCID, depending on the specific configuration of the ground station.
\[
\begin{align*}
MCID & = TFVN + SCID; \\
GVCID & = MCID + VCID \\
& = TFVN + SCID + VCID.
\end{align*}
\]

Figure 9 - Satellite Identification Relations [2]

2.5.2.2. Transmission Control

Communications Operation Procedure-1 (COP-1) is implemented in the TC space data link protocol “to enable the delivery of service data units to the receiving end of the layer above, correct without omission or duplication, and in the same sequential order in which they were received from the layer above at the sending end.” [15]

COP-1 makes use of two additional protocols “a Frame Operation Procedure-1 (FOP-1) that executes within the sending entity, and a Frame Acceptance and Reporting Mechanism-1 (FARM-1) that executes within the receiving entity. The FOP-1 transmits Transfer Frames of a particular Virtual Channel to the FARM-1 of the same Virtual Channel. The FARM-1 returns reports of the status of Transfer Frame acceptance to the FOP-1 using the Communications Link Control Words (CLCWs).” [15]

2.5.2.3. TC Transfer Frame

The Telecommand transfer frame is a contiguous set of fields that are transferred to the spacecraft. The three main components that make up this transfer frame are shown below in Figure 10 and explained in detail in the next sections. The primary header and data fields are mandatory, while the frame error control field is optional and applied depending upon the mission configuration.
Note that the above figure lists “Up to 1019 octets,” meaning that the value could be lesser depending upon some other mission characteristics, such as additional headers or protocols embedded in the data field.

### 2.5.2.4. TC Primary Header

“The Transfer Frame Primary Header is mandatory and shall consist of eight fields, positioned contiguously, in the following sequence:

- a) Transfer Frame Version Number (2 bits, mandatory);
- b) Bypass Flag (1 bit, mandatory);
- c) Control Command Flag (1 bit, mandatory);
- d) Reserved Spare (2 bits, mandatory);
- e) Spacecraft Identifiers (10 bits, mandatory);
- f) Virtual Channel Identifier (6 bits, mandatory);
- g) Frame Length (10 bits, mandatory);
- h) Frame Sequence Number (8 bits, mandatory);” [2]

A visual depiction of this frame can be seen in Figure 11.
The transfer frame version number shall always be ‘00’ to signify the TC frame depicted above. The bypass flag determines the type of transfer frame, either A (‘0’) or B (‘1’). Type-A signifies that the Frame Acceptance and Reporting Mechanism (FARM) in the COP shall operate normally, while type-B enables a bypass of this mechanism. The control command flag is similar to the bypass flag in that it determines if the frame is type-C (‘1’) or type-D (‘0’). Type-C indicates that the payload is control information, while type-D signifies a Frame Data Unit (FDU).

The SCID and the VCID have been discussed in the sections above. The frame length indicates the length of the entire TC frame including headers, trailers, the PDU, and the FECF if included. The frame sequence number is utilized for type-A frame by the COP and FARM.

2.5.2.5. Frame Error Control Field

The Frame Error Control Field (FECF) is a 16-bit Cyclic Redundancy Check (CRC) across the entire frame, although the FECF utilizes a non-standard polynomial to do this check. This must be verified upon receipt of a TC frame to validate transmission success. No bits may be modified prior to this check or the result will be invalid. A large amount of reference material and code is available to further explain this technique. The logic diagram of an example decoder is displayed in Figure 12.
2.5.2.6. TC Security

“Additional fields are defined for a TC Transfer Frame to support the use of the SDLS security features in the frame.” “There are two fields for the SDLS protocol: a Security Header and a Security Trailer.” [2] A comparison of TC frames with and without the SDLS fields can be seen below in Figure 13. It should be noted that the utilization of these additional fields effectively limits the maximum payload data length possible in a single transfer frame by the size of the fields. The fields in each of these additional security blocks are discussed in the Section 2.5.4 - Space Data Link Security.
2.5.3. Telemetry

The Telemetry (TM) Protocol encapsulates the downlink from the spacecraft to the ground station, or the return space link. This protocol shares many similarities with TC, except that it does not provide mechanisms for retransmission, as that occurs only on the receiving end of these protocols.

2.5.3.1. TM Transfer Frame

The TM frame has an additional trailer when compared to the TC frame. The Operational Control Field (OCF) includes reports as to the status of the TC protocol. This field is optional and determined by the mission configuration, but must be located immediately following the data field and precisely 4 bytes in size. The TM transfer frame can be seen below in Figure 14. The following sections will break down the headers as they are needed in the context of the implemented Crypto library.
2.5.3.2. TM Frame Primary Header

“The Transfer Frame Primary Header is mandatory and shall consist of six fields, positioned contiguously, in the following sequence:

a) Master Channel Identifier (12 bits, mandatory);
b) Virtual Channel Identifier (3 bits, mandatory);
c) Operational Control Field Flag (1 bit, mandatory);
d) Master Channel Frame Count (1 octet, mandatory);
e) Virtual Channel Frame Count (1 octet, mandatory);
f) Transfer Frame Data Field Status (2 octets, mandatory).” [16]

The master channel identifier can further be broken into the Transfer Frame Version Number, the same use and value of ‘00’ as the TC frame, and the assigned SCID. A ‘1’ for the OCF flag indicates the expectation of the 4 OCF bytes at the end of the transfer frame data field. The master and virtual channel counters both increment in the Crypto library as the return channels are static. Finally, the data field status makes up the remaining two bytes of the primary header as seen in Figure 15.
2.5.3.3. **TM Data Field Status**

“This 16-bit field shall be sub-divided into five sub-fields, as follows:

a) Transfer Frame Secondary Header Flag (1 bit, mandatory);
b) Synchronization Flag (1 bit, mandatory);
c) Packet Order Flag (1 bit, mandatory);
d) Segment Length Identifier (2 bits, mandatory);
e) First Header Point (11 bits, mandatory).” [16]

The transfer frame secondary header flag is set to ‘0’ in the Crypto library and further discussion of the secondary header will be omitted, as it is outside the scope of this paper. The synch flag “shall be ‘0’ if octet-synchronized and forward-order Packets or Idle Data are inserted; it shall be ‘1’ if a VCA_SDUs is inserted.” [16] This value is set as a static ‘0’ since the Crypto library does not support VCA_SDUs at the writing of this paper. Due to the synch flag being set, both the packet order flag and segment length ID are also statically set to zero. The first header pointer sub-field allows for an offset of the data in the frame data field in case a packet was already being loaded with idle data when true telemetry was provided. A depiction of the data field status is provided below in Figure 16.
2.5.3.4. Operations Control Field

The Operations Control Field (OCF) allows for telemetry of the uplink or TC to be communicated. This type of frame is known as the Communications Link Control Word (CLCW) and is discussed in this section because it is contained in the TM frame, although it relates directly to the TC and COP operational states. The CLCW is discussed in detail in the TC Protocol documentation, but in short “the CLCW shall consist of ten fields, positioned contiguously, in the following sequence:

a) Control Word Type (1 bit, mandatory);
b) CLCW Version Number (2 bits, mandatory);
c) Status Field (3 bits, mandatory);
d) COP in Effect (2 bits, mandatory);
e) Virtual Channel Identification (6 bits, mandatory);
f) Reserved Spare (2 bits, mandatory);
g) Flags (5 bits, mandatory);
h) FARM-B Counter (2 bits, mandatory);
i) Reserved Spare (1 bit, mandatory);
j) Report Value (8 bits, mandatory).” [2]

Currently the COP in Effect bits are set to ‘00’ in the Crypto library, as this additional protocol has not yet been implemented. The virtual channel identification will also be static due to a single channel expected in this implementation. The remainder of the sub-fields are utilized by COP and are zeroed. These sub-fields, in order, can be seen in Figure 17.
2.5.4. Space Data Link Security

“The purpose of the Security Protocol is to provide a secure standard method, with associated data structures, for performing security functions on octet-aligned user data within Space Data Link Protocol Transfer Frames over a space link.” [6] The security header and trailer are a simple addition, cutting into the total data field length, to transfer frames in order to provide security. This section details each of the specific transfer frame types and the security that they provide. Figure 18 below provides an example of how this security protocol is implemented.
The SDLS protocol resides in the Data Link Protocol Sublayer within the OSI Basic Reference Model.

### 2.5.4.1. Security Header

“The Security Header shall consist of one mandatory field and three optional fields, positioned contiguously, in the following sequence:

a) Security Parameter index (16 bits, mandatory);

b) Initialization Vector (octet-aligned, fixed-length for the duration of the SA, optional);

c) Sequence Number (octet-aligned, fixed-length for the duration of the SA, optional);

d) Pad Length (octet-aligned, fixed-length for the duration of the SA, optional).

A Security Header shall consist of less than or equal to 64 octets.” [6]

### 2.5.4.2. Security Trailer

“The Security Trailer, if present, shall consist of a MAC (octet-aligned, fixed-length for the duration of the SA).” [6] The MAC is essentially a field containing the CMAC across the encrypted frame and is
the only data provided in the security trailer. The option exists to allow calculation utilizing Additional Authenticated Data (AAD) with an Authentication Bit Mask (ABM) applied. The additional data is typically the SDLS primary and security header, which is never encrypted. The security trailer is shown below in Figure 20. Note that this only appears when in a security mode, not clear.

![Security Trailer](image)

Figure 20 - Security Trailer [6]

“CMAC is a keyed hash function that is based on a symmetric key block cipher such as AES. Cipher-based authentication uses the properties of a block cipher algorithm, rather than a hash function, to create a MAC.” [17] This type of MAC, opposed to a standard CRC, provides for the detection of unauthorized data modification as it encompasses the entire frame. The generation of the CMAC is depicted below in Figure 21.

![CMAC Generation](image)

Figure 21 – CMAC Generation [17]

2.5.4.3. Security Associations

“The Security Protocol provides security associations for defining the cryptographic communications parameters to be used by both the sending and receiving ends of a communications session, and for maintaining state information for the duration of the session. A Security Association (SA) defines a simplex (one-way), stateful cryptographic session for providing authentication, data integrity, replay
protection, and/or data confidentiality.” [6] The SA contains the type of service to be provided: clear, authentication only, encryption only, or authenticated encryption. Additional configuration is stored in each SA.

“Both the sender and the receiver must create an SA, associate it with cryptographic key(s), and activate it before the SA may be used to secure Transfer Frames on a channel. SAs may be statically preloaded prior to the start of a mission. SAs may also be created dynamically as needed, even while other existing SAs are active. The mechanism for switching from one active SA to another is an Application Layer function.” [6]

“It is possible to create a ‘clear mode’ SA using one of the defined service types by specifying the algorithm as a ‘no-op’ function (no actual cryptographic operation to be performed). Such an SA might be used, e.g., during development testing of other aspects of data link processing before cryptographic capabilities are available for integrated testing.” [6]

2.5.4.4. Authentication

“The Security Protocol provides for the use of authentication algorithms to ensure the integrity of transmitted data and the authenticity of the data source. The Security Protocol also provides for the use of sequence numbering to detect the unauthorized replay of previously transmitted data.” [6] These authentication algorithms include a MAC computed over the encrypted data and a FECF calculated across the entire frame.

“When the Security Protocol is used for authentication, a sequence number is also transmitted in the Transfer Frame. As part of an SA providing authentication, both the sender and the receiver manage the follow information:

a) A sequence number value (current value for the sender, expected value for the receiver);

b) A sequence number windows for comparison by the receiver;

c) The location within the Transfer Frame of the sequence number.” [6]

2.5.4.5. Encryption

“The Security Protocol provides for the use of encryption algorithms to ensure the confidentiality of transmitted data. When the Security Protocol is used for encryption, that data area of the frame (the ‘plaintext’) is replaced with an encrypted version of the same data (the ‘ciphertext’). An initialization
vector is often used as an input to the encryption process. Depending upon the cryptographic algorithm and mode used, additional fill data may be needed to pad any undersized blocks.” [6]

2.6. Galois Counter Mode AES

AES/GCM (Galois/counter mode) is the recommended standard for authenticated encryption by the CCSDS Cryptographic Algorithm Blue Book [9]. This is the only means of encryption currently supported by the Cryptography Library. Counter mode allows for each block to be independent of one another, allowing for encryption and decryption to work in parallel without the additional overhead of padding blocks necessary for other modes.

“Counter mode differs from other encryption modes because the plaintext data to be encrypted is not directly run through the AES algorithm. Rather, a counter which has been combined with a cipher key is used as the starting input to the algorithm. This produces 128-bit random data blocks. The block bits are XORed with the plaintext data to produce the output cipher blocks.” [17] However, the counter must not be repeated to avoid compromising the unique key. A summarized diagram of the GCM AES encryption technique is provided in Figure 22 and additional details can be found in NIST Special Publication 800-38D.

![Galois Counter Mode AES Diagram](image.png)

Figure 22 - Galois Counter Mode AES Diagram [17]
2.7. Space Data Link Security – Extended Procedures

“The purpose of the SDLS Extended Procedures is to provide a standardized set of auxiliary services that are necessary to operate an implementation of the SDLS protocol.” [4] The three types of services are detailed in the following sections. In addition to these three, a Frame Security Report (FSR) is also specified. This report provides insight into the current state of the Cryptography Library and alternates in the OCF with the CLCW. All of these services build upon the core protocol.

2.7.1. Key Management Service

“The proper management of cryptographic keys is essential to the effective use of cryptography in security. Keys are analogous to the combination of a safe. If an adversary knows a safe combination, the strongest safe provides no security against penetration. Similarly, poor key management may easily compromise strong algorithms. Ultimately, the security of information protected by cryptography directly depends on the strength of the keys, the effectiveness of mechanisms and protocols associated with the keys, and the protection afforded to the keys. All keys need to be protected against unauthorized disclosure. Key Management provides the foundation for the secure generation, storage, distribution, use and destruction of keys.” [8]

Two types of keys exist in the standard, Master and session keys. Master keys are used only for the uploading of new session keys via the over-the-air-rekeying (OTAR) procedure. Session keys are to be utilized for all other applications. Both types of keys follow the same lifecycle as shown in Figure 23. A key can be reset to the Pre-Activation state via an OTAR command from any previous state allowing for re-use of the memory space in which a key is located. Additional transition rules are detailed in the standard.

![Figure 23 – Cryptographic Key Lifecycle [4]](image)

Additional states such as suspended and compromised are optional and not included in Figure 23 above, as these states are mission-specific and have not been included in the initial release of the
Cryptography Library. Additional functions such as key verification and status request are expanded upon in the standard.

2.7.2. Security Association Management Service

The SA management service provides the bare necessities for complex mission operations to “command the configurable Security Association parameters of a remote system’s SDLS implementation into a state suitable for operations”. [4] A complex mission would include those that need to create new associations and re-key them once on orbit. Not all missions will be complex, but long duration missions may need this feature. The state machine for SAs is captured in Figure 24.

![Variable State Model for Security Association Management](image)

Figure 24 - Variable State Model for Security Association Management [4]

SAs are mapped to Global Virtual Channel and/or Global MAP IDs. An SA can be mapped to multiple GVCIDs, but a single GVCID can only map to one SA. This mapping occurs in the SA Start function, changing the current SA state from keyed to operational.

2.7.3. SDLS Monitoring and Control Service

The SDLS monitoring and control service allows for logging, reporting, and determining of the current status of the Cryptography Library. This service also provides a means to debug errors that occurred and reset alarms that have been triggered “allow(ing) for complete control/command of the on-board security processor(s)”. [4]
2.7.4. Extended Procedures Payload Data Unit

The SDLS-EP services can be accessed via utilization of the PDU in Figure 25. Note that this message is contained in the data field of an SPP packet, defined in Section 2.7.6, and includes the extended procedures tag, length, and value, or TLV. This TLV formula is interpreted for each of the commands accepted by the Cryptography Library to allow for the determination of the data. Specifically, the T or Tag in TLV is broken down into four sub-fields shown in Figure 25. The first bit contains the type field which defines if the packet is a command or a reply, while the second bit is used as a user flag to allow additional commands not dictated by the standard to be implemented. The service group decides which of the three services the packet relates to while the procedure identifier (PID) specifies the exact item. The listings for all defined, non-user, PIDs is captured in Table 5.

![Extended Procedures PDU](image)

**Figure 25 - Extended Procedures PDU [4]**

<table>
<thead>
<tr>
<th>Procedure Identification</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>Over-The-Air-Rekeying (OTAR)</td>
</tr>
<tr>
<td>0010</td>
<td>Key Activation</td>
</tr>
<tr>
<td>0011</td>
<td>Key Deactivation</td>
</tr>
<tr>
<td>0100</td>
<td>Key Verification</td>
</tr>
<tr>
<td>0101</td>
<td>Key DB Status Request*</td>
</tr>
<tr>
<td>0110</td>
<td>Key Destruction</td>
</tr>
</tbody>
</table>

*Not currently implemented in the Cryptography Library

**Table 3 - EP Key Service Identifications**
### Procedure Identification Assignment

<table>
<thead>
<tr>
<th>Assignment</th>
<th>0000</th>
<th>Delete SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expire SA</td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td>Stop SA</td>
<td>0010</td>
<td>Create SA</td>
</tr>
<tr>
<td>Rekey SA</td>
<td>0110</td>
<td>Start SA</td>
</tr>
<tr>
<td>SA Status Request</td>
<td>1000</td>
<td>Set Anti-Replay Counter</td>
</tr>
<tr>
<td>Set Anti-Replay Counter Window</td>
<td>1001</td>
<td>Set Anti-Replay Counter Window</td>
</tr>
</tbody>
</table>

Table 4 – EP SA Service Identifications

<table>
<thead>
<tr>
<th>Assignment</th>
<th>0001</th>
<th>Ping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log Status Request</td>
<td>0010</td>
<td>Dump Log</td>
</tr>
<tr>
<td>Erase Log</td>
<td>0011</td>
<td>Self-Test*</td>
</tr>
<tr>
<td>Read Sequence Number</td>
<td>0100</td>
<td>Reset Alarm Flag</td>
</tr>
<tr>
<td></td>
<td>0110</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0111</td>
<td></td>
</tr>
</tbody>
</table>

Table 5 – EP MC Service Identifications

#### 2.7.5. Frame Security Report

“This Frame Security report contains information about the status of the security unit and about the security processing (e.g. indicating a recent authentication failure).” [4] The alarm field is flagged upon any error occurring. Additional fields are included to quickly determine the type of error that occurred. These flags are not cleared until the receipt of an alarm reset command. The Frame Security Report (FSR), as seen in Figure 26, also includes the last SPI and sequence number or initialization vector utilized in order to sync the spacecraft and the ground station in the event of RF link failure.
2.7.6. Space Packet Protocol

“The Space Packet Protocol is designed to meet the requirements of space missions to efficiently transfer space application data of various types and characteristics over a network that involves a ground-to-space or space-to-space communications link”. [18] The efficiency and dual purpose is the reason this protocol is the base layer wrapped up by both the TC and TM protocols. “The protocol entity performs the following protocol functions:

a) Generation (or validation) and processing of protocol control information included in the header to perform data identification;
b) Routing of protocol data units through a series of underlying subnetworks;
c) Multiplexing/demultiplexing in order for various service users (i.e., various LDPs) to share a logical connection provided by an underlying subnetwork.” [18]

2.7.6.1. Space Packet Frame

“A Space Packet shall encompass the major fields, positioned contiguously, in the following sequence:

a) Packet Primary Header (6 octets, mandatory);
b) Packet Data Field (from 1 to 65536 octets, mandatory).” [18]

Figure 27 below displays the Space Packet Frame and a view of the high level fields.
The secondary packet header has multiple types that are utilized depending upon the message type, further detailed in the next section. Additional mission specific headers are possible.

2.7.6.2. Packet Primary Header

“The Packet Primary Header is mandatory and shall consist of four fields, positioned contiguously, in the following sequence:

a) Packet Version Number (3 bits, mandatory);

b) Packet Identification Field (13 bits, mandatory);

c) Packet Sequence Control Field (16 bits, mandatory);

d) Packet Data Length (16 bits, mandatory).” [18]

Figure 28 below depicts the sub-fields in the Space Packet Protocol Primary Header.
The packet version number for this standard is set to ‘000’. The packet type is set to ‘1’ for commanding and ‘0’ for telemetry. The secondary header flag is always enabled per flight software requirements. The application identifier allows for the destination of the packet to be known. The sequence flags are statically set to ‘11’ meaning that the packet contains unsegmented User Data, due to the mission configuration. The packet sequence count is simply another counter to ensure that a packet is not lost in translation. Packet data length provides the length of the packet data field minus one, which includes both the packet secondary and user data fields described in the next sections.

2.7.6.3. Packet Secondary Header

Out of the box, the Core Flight System (cFS) includes separate secondary headers for commands and telemetry specifying the command or providing the time at which the telemetry item was generated. The official protocol specifies Figure 29 as the secondary header, but leaves the fields variable to allow for mission specific options.
2.7.6.4. Packet Data Field

The ESA has additional fields contained in the data field that are typically not utilized by NASA. These fields are included in the Cryptography Library to increase interoperability and ease testing with the ESA. The first addition is the Packet Utilization Standard (PUS), which could be another secondary header. The sub-fields of the PUS, shown in Figure 30, are not utilized, but are read by the Cryptography Library and stored in the event that these become necessary for future missions.
The additional packet error control field is also utilized by the ESA and not captured in the CCSDS standard. This field is simply a 16-bit CRC across the entirety of the packet. This CRC is not checked by the Cryptography Library for error, but is appended to all messages generated as a reply to a command.

Figure 30 - Packet Data Field Diagram [19]
CHAPTER 3 | DESIGN

3.1. Starting Point

NASA IV&V has performed inter-agency point-to-point interoperability testing with CNES for the CCSDS Security Working Group in the past. This initial testing was for the Space Data Link Security - Core Protocol, discussed above, which could not be published until fully tested. Specifically, the AOS protocol on the downlink side was tested. The designed approach utilized the existing GPM Operational Simulator (GO-SIM) developed by the JSTAR - ITC Team. This simulator emulates the spacecraft down to the flight hardware level while actually running a flight software binary, allowing for a full flight-like environment to run scenarios and test in. [20]

“In an effort to reduce code duplication, a simple cross-platform security library was developed to be used in both the ground and spacecraft systems. Due to limited memory on the spacecraft hardware, the library was to be kept as compact as possible.” This initial library supported the following per the SDLS-CP baseline testing:

- Advanced Encryption Standard (AES) algorithm
- Cipher-Based Message Authentication Code (CMAC) for authentication
  - Based on AES algorithm with 128-bit keys
  - 32-bit Anti-Reply Sequence Number
  - 128-bit output Message Authentication Code (MAC)
- Galois Counter Mode (GCM) for authenticated encryption
  - Based on AES algorithm with 128-bit keys
  - 96-bit Initialization Vector
  - 128-bit output MAC

Due to complications between agencies and test beds, a direct connection for interoperability testing was not feasible. Instead, secure email transactions delivered initial configuration followed by the frames for which to test with. The responses to these tests were recorded and compared.
3.1.1. Previous Configuration

The GO-SIM spacecraft simulator utilizes cFS and the ASIST ground station, allowing full mission operations to take place. The FSW runs on the Simics hardware emulator platform provided by Wind River which permits the pausing and running of flight software binaries that are compiled for the actual spacecraft, allowing for the injection of or monitoring for errors at the register level and above. Additionally, the buses are simulated to permit communications to various hardware models. A diagram of this can be seen in Figure 31. A minimal library was embedded inside of FSW applications and the ASIST ground station to allow for the processing of commands and telemetry per the SDLS protocol.

![Diagram of previous work architecture](image)

Figure 31 - Previous Work Architecture Diagram [20]

3.1.2. Previous Approach

“The data (telemetry and telecommand) is exchanged in real time. Encrypted telecommands will be sent to the FSW via FEDS and an SCOMM simulator. The incoming data can then be decrypted by the CI application within the FSW. Telemetry, sent from the spacecraft to the ground, functions the same way but in reverse. The TO application within the FSW encrypts the telemetry and downlinks via the SCOMM simulator to FEDS and then decrypted by FEDS and passed to the ground station ASIST.”
The data flow for both uplink and downlink follow similar paths but utilize different applications inside the flight software, as seen in Figure 32 below.

![Figure 32 - Previous Work Data Flow](image)

To support SDLS on the telemetry side, modifications were made to the Front End Data System (FEDS) component of ASIST and the flight software. The flight software utilized cFS and VxWorks as the operating system. A lightweight library was created in cFS to handle the cryptography and specifically included the GCM AES 128 protocol and CMAC for both encryption and decryption.

### 3.1.3. Local Testing

The unit tests used to verify SDLS-CP for this previous work are still available in the Cryptography Library. The ability to run the lightweight AES implementation for both encryption and decryption is covered, along with the GCM which utilizes a provided key, IV, and AAD. These tests run off of log files. The log files contain the input, expected output, and expected result of the test. The following bullets are to be executed in order to produce the results shown in Figure 33 and Figure 34.

- cd ~/cFS_SDLS/apps/crypto/unit_tests
- make
- ./itc_aes128_test
3.1.4. Inter-Agency Testing

The ITC Team generated 24 dummy frames and secured the frames using the baseline recommendation for AOS in the telemetry link. Some frames were modified to purposely raise faults. CNES processed the secure frames and sent the results back to the ITC Team for verification, a depiction of which can be seen in Figure 35 below.
Total success was achieved on the first attempt. The independent implementations produced identical and accurate results, validating the accuracy of the standard. Due to this success, the SDLS protocol could be published as a Blue Book Recommended Standard. [6]

### 3.2. High Level Design

Even during the previous work, ideas were being formulated for a more robust system and initial protocol implementations were received for the SDLS-EP review. Upon completion, an opportunity arose to refactor the lightweight Cryptography Library into one which promoted re-use, simple deployment, and ease of understanding while also verifying the SDLS-EP protocol implementation. This section details the requirements imposed upon this work along with all the components involved, the interaction between them, and the means by which testing is to be performed.

#### 3.2.1. Components

By documenting all pieces necessary for this new effort, the search for appropriate applications that meet certain criteria can begin. As in the previous work, two sides exist. One side is the mission control system and both the beginning and end of the information transfer. Commands are sent from here or another authorized mission control center to a ground station. The ground station handles all of the complexities of RF communications and the closing of the space link to the satellite or spacecraft. The
spacecraft must then interpret the command and perform the desired function. For the purpose of proving the SDLS-EP protocol and the Cryptography Library, a simulation of the spacecraft must exist for testing that mimics a system that could actually fly. These paths listed have been included in Figure 36 and show where the security unit, or Cryptography Library in this case, would be involved. The concept of test as you fly and fly as you test should be upheld.

![Diagram of ESA SDLS-EP Testing Architecture](image-url)

Figure 36 – ESA SDLS-EP Testing Architecture [21]

The ESA SDLS-EP Testing Architecture has arrows in both directions, but for the selected protocols to be utilized for testing they can each only operate in a defined direction. The protocols to be used are TC and TM. The mission control system will be generating the TCs containing the commands while the spacecraft will be interpreting or processing them. In the same fashion, the spacecraft will be generating responses and data to be returned to the sender to handle. The SDLS-EP protocol is only to be used during communication of the frames, for example from sender to receiver. Once received and processed, the protocol is not in use until the data is ready to be sent. In Figure 37, the lines in which the protocol is active are considered protected and are shown in red while others are shown in black.
3.2.2. Cloud Testing

Virtualization in the cloud was also selected to be utilized for testing this scenario. Cloud Sigma was chosen due to being pre-approved by the ESA for cloud operations. NASA and the ESA each purchased accounts and deployment isolated environments to preserve the independence of the agencies and avoid legal challenges. To connect the two virtual machines, a virtual local area network (VLAN) was utilized, as diagrammed in Figure 38. “This setup is very beneficial in terms of security since access to the VLAN from outside is strictly limited to whitelisted IP addresses.” [21] Due to the VLAN configuration, static IP addresses were required and purchased via the service provider.
The use of the cloud for testing permits much more comprehensive testing compared to the previous work. Testing can occur in real time and in a more realistic environment that allows for additional errors or unexpected events to drive the need for better understanding and clarification of the protocol under test.

### 3.2.3. Requirements

Per the previous sections, these are the derived requirements of the Cryptography Library:

- Compatibility with SDL protocols
- Compatibility with SLE services
  - TC / TM / AOS
- Modularity / Flexibility
- Independence of cryptographic algorithms
- Interoperability
• Decoupling of SDI and SDLS data integrity performance
• Space Data Link Security – Extended Procedures Baseline Mode
• Remove need for ASSIST ground station / FEDS
• Remove need for major application additions for inclusion of library
• Space Link Extension support
• Straightforward deployment

3.3. Software Details

3.3.1. Core Flight System

The Core Flight System (cFS) was developed by GSFC and has been utilized by a multitude of space missions. The flight heritage extends over 15 years and is growing daily. cFS provides a base set of applications originally called the Core Flight Executive (CFE). This provides a software bus and scheduler along with time, event, and table services to all other applications. cFS also enables the use of an “app store” where users can pick and choose or develop independent applications and libraries necessary for a particular mission. Both the core and application layers sit on an abstraction layer. This layer allows for compilation on a variety of operating systems and for various hardware. Details about each of the components are in the next two sections.

3.3.1.1. Operating System Abstraction Layer

The Operating System Abstraction Layer (OSAL) officially supports Linux, RTEMS, and VxWorks and unofficial support for FreeRTOS. This enables flight software developers to compile and test in a Linux environment while being able to build in a flight configuration without modification to high-level applications or libraries. All development and testing of the Cryptography Library has utilized a Linux build.

3.3.1.2. Platform Support Package

“The PSP is a software library that contains all of the software needed to adapt the CFS to a particular processor card. The PSP also provides an API to an abstract processor hardware computing system including hardware devices such as memory, I/O ports, and non-volatile memory.” [23] The PSP
enables porting to a variety of hardware as well. This layer could also be referred to as a Hardware Abstraction Layer.

### 3.3.2. Space Link Extension Software

The JSTAR ITC Team typically includes the ground station software in spacecraft simulators in order to easily command the spacecraft and utilize the product for training operators. Due to the focus on the spacecraft and emulating the hardware, the connection from the ground station and the spacecraft itself is typically a bent pipe to avoid unnecessary simulated components. The SLE protocol has been implemented for various missions, but has never been deployed at the NASA IV&V facility during any previous work. In fact, no subject matter experts could be found at the facility on SLE or CCSDS protocols in general.

Due to ESA requirements, the SLE protocol would also have to be understood and deployed in order to proceed with testing. The JSTAR ITC Team initially had access to an SLE program via the NASA Technology Transfer process. This package provided simplistic functions to utilize the provided ASN.1 API that is readily provided with the SLE standard provided by CCSDS. The ASN.1 package required an additional license fee, which motivated the team to search for other potential options. During this search, a software package utilized in a James Webb Space Telescope simulation and testing environment at GSFC was discovered. This software, however immature, provided enough stability to allow for proper connection and communication with the ESA counterpart.

This software package was released to IV&V unnamed, but has since been referred to as SLE_IVV. The package was written in C++ and required the most recent versions of GCC and another mothballed software, CppUnit, for compilation. Creating the environment for this was extremely difficult and severely delayed testing and development of the Cryptography Library, as documentation for this software was non-existent and the code was illegible. With the assistance of other JSTAR ITC engineers, the requested ESA configuration for interoperability was embedded and the actual flow of data was understood. The high-level diagram can be seen in Figure 39.
3.3.3. Python Connectors

Due to the SLE_IVV program requiring a TCP/IP connection and the chosen cFS application requiring UDP, additional connectors were needed to join the FSW with the SLE program. Python was chosen due to the speed at which modifications could be made for testing and the quick turnaround time. These connectors also allowed for modification of the data per unique operational requirements of the SLE software, as well as the ESA simulator.

The Command Ingest (CI) SLE Link script accepts SLE frames from the SLE_IVV software while performing modifications and checks on the data prior to forwarding it along to the FSW, specifically the CI Lab application. The lab application, when compared to the standard one, allows for connection via UDP opposed to a standard UART / I2C / SPI which would be utilized for an actual radio. The connector itself also allows for recovery of a failing in the SLE_IVV software. This failing is because occasionally the SLE header and payload are transmitted in separate packets, which causes unnecessary errors to be triggered due to missing or improperly formatted data when forwarded to the FSW.

Upon testing, it was noted that the ESA SLE program appends additional data every eighth byte. This data is assumed to be a CRC and is not described by the CCSDS SLE protocol, nor was it known to the ESA tester as the simulator used for testing locally removes this data prior to processing.

The Telemetry Output (TO) SLE Link script provides similar functionality, but in reverse by accepting TM frames from the TO Lab application via UDP and forwarding them to the SLE_IVV program. This link also provides the option to enable telemetry downlink from the TO Lab application if it is not enabled by default in FSW. Additional checks and manipulation of the data are done to ensure that it meets the ESA interoperability mission specifications and SLE_IVV requirements, including static data lengths for packets and an additional SLE - RCF header necessary for processing.
3.3.3.1. User Commands

An additional script was developed to enable the sending of specified user commands for testing. Upon running, a menu is displayed that provides options for the desired command to be sent to the FSW and interpreted by the Cryptography Library. This menu is shown in Figure 40. Additional options are provided in menu format for selections that require them, such as the modify commands.

![User Commands Script](image)

Figure 40 - User Commands Script

3.4. Architecture

Using all components elaborated upon in the previous sections, the overview diagram in Figure 41 can be assembled. The same Cryptography Library, although different functions of it, is utilized in each direction. Upon receipt of a command by the CI Lab app, it is decrypted per the security association referenced in the header. Instead of simply forwarding all messages to the software bus, a check for a message bound for the Cryptography Library is immediately performed. All messages bound for other destinations determined by application ID in the SPP header are put on the bus for proper routing.
3.5. Deployment

3.5.1. cFS Application Modifications

Only two applications were selected for use during testing, CI Lab and TO Lab. The Lab version implies that communication is done over UDP with no regard to protocols such as TC, TM, and COP-1 that are typically implemented. The standard use for lab applications is sending software bus messages.
to applications for testing purposes or monitoring the messages on the bus of specific types. Modifications were made to each of these applications in order to integrate the Cryptography Library.

### 3.5.1.1. CI Lab

```c
// Process Security - START
result = Crypto_TC_ProcessSecurity((char *) &CI_IngestBuffer, &status);
if (result != OS_SUCCESS)
{
    // TODO: Crypto Process Security Error
    OS_printf("ci_lab_app: Crypto_TC_ProcessSecurity Error!\n");
    return;
}
else
{
    if (status == 0)
    {
        return;
    }
}
// Process Security - END
```

### 3.5.1.2. TO Lab

```c
... // Process Security - Variable

// Apply Security - START
CFE_PSP_MemCpy(ingest, PktPtr, size);
status = Crypto_TH_ApplySecurity((char*) ingest, (int*) &size);
// Apply Security - END
if (status == OS_SUCCESS)
{
    status = sendto(TLMsockid, (char *)ingest, size, 0, 
                     (struct sockaddr *) &s_addr, 
                     sizeof(s_addr));
}
```

### 3.5.2. Initial Configuration

Currently, in the Cryptography library, the initial configuration is hard coded into the library. Upon boot, the FSW runs a “crypto_Init” function, which initializes and configures security associations, master and session keys, operational control fields, CRC tables, and the telemetry frame. At present, this allows every reset to return the Cryptography Library to the default configuration, providing each of the tests to have a fundamental starting point. For actual mission deployment, a power-on reset
would read in the most recent configuration from non-volatile memory. Updates to configuration via the SDLS-EP services would be recorded in non-volatile memory to preserve the state.

3.6. Operations

3.6.1. Baseline Mode

The baseline mode is used to describe the minimum functionality for operations per the given standard. This is also the level at which agencies involved in interoperability testing are required to achieve in order to publish the standard. The Cryptography Library was officially evaluated utilizing SDLS-EP baseline mode utilizing TC and TM without the inclusion of the COP-1 protocol, but additional interoperability testing was performed to verify the functionality found in intra-operability testing, get to desired states for testing, and gain general understanding of the standard.

3.6.2. Mission Specific Options

The ‘crypto_config.h’ file contains mission-specific options that are subject to change per mission requirements. These options, along with simple definitions for states to increase code readability, are defined in this file. The most basic and generic options are elaborated upon in Table 6.
<table>
<thead>
<tr>
<th>Field</th>
<th>Default Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCID</td>
<td>3</td>
<td>Space Craft ID</td>
</tr>
<tr>
<td>NUM_GVCID</td>
<td>64</td>
<td>Number of Global Virtual Channel IDs maintained in memory.</td>
</tr>
<tr>
<td>KEY_SIZE</td>
<td>16</td>
<td>Maximum key size available</td>
</tr>
<tr>
<td>KEY_ID_SIZE</td>
<td>8</td>
<td>Bits</td>
</tr>
<tr>
<td>NUM_KEYS</td>
<td>256</td>
<td>$2^{\text{KEY_ID_SIZE}}$</td>
</tr>
<tr>
<td>IV_SIZE</td>
<td>12</td>
<td>Maximum Initialization Vector size available</td>
</tr>
<tr>
<td>OCF_SIZE</td>
<td>4</td>
<td>Operational Control Field Size (TM)</td>
</tr>
<tr>
<td>MAC_SIZE</td>
<td>16</td>
<td>Message Authentication Code size</td>
</tr>
<tr>
<td>FECF_SIZE</td>
<td>2</td>
<td>Frame Error Control Field size, bytes</td>
</tr>
<tr>
<td>ECS_SIZE</td>
<td>4</td>
<td>Encryption Cipher Suite size</td>
</tr>
<tr>
<td>ABM_SIZE</td>
<td>20</td>
<td>Maximum Authentication Bit Mask size</td>
</tr>
<tr>
<td>ARC_SIZE</td>
<td>20</td>
<td>Maximum Anti-Replay Counter size</td>
</tr>
<tr>
<td>ARCW_SIZE</td>
<td>1</td>
<td>Maximum Anti-Replay Counter Window size</td>
</tr>
<tr>
<td>SN_SIZE</td>
<td>0</td>
<td>Sequence Number size</td>
</tr>
<tr>
<td>EMV_SIZE</td>
<td>2</td>
<td>Event Message Value size</td>
</tr>
<tr>
<td>LOG_SIZE</td>
<td>256</td>
<td>Maximum number of logs in memory</td>
</tr>
<tr>
<td>TC_SH_SIZE</td>
<td>8</td>
<td>Segment Header size</td>
</tr>
<tr>
<td>TC_SN_SIZE</td>
<td>0</td>
<td>Sequence Number size</td>
</tr>
<tr>
<td>TC_PAD_SIZE</td>
<td>0</td>
<td>Padding size</td>
</tr>
<tr>
<td>TC_FRAME_DATA_SIZE</td>
<td>1740</td>
<td>Maximum frame data size</td>
</tr>
<tr>
<td>TLV_DATA_SIZE</td>
<td>494</td>
<td>Maximum SDLS-EP data size</td>
</tr>
<tr>
<td>TM_FRAME_DATA_SIZE</td>
<td>1740</td>
<td>Maximum TM frame data size</td>
</tr>
<tr>
<td>TM_FILL_SIZE</td>
<td>1145</td>
<td>Defined fill size of the TM frame</td>
</tr>
<tr>
<td>TM_PAD_SIZE</td>
<td>2</td>
<td>Padding size in bytes</td>
</tr>
</tbody>
</table>

Table 6 - Mission Specific Options Defined

Additional fields for build type, static or dynamic, and functionality options also exist. Typically, FSW that will fly in space will be built statically, but for testing a dynamic build was utilized. The option
to fill a TM frame to the specified size also exists, along with the type of fill, either zero or with idle packets defined by the ESA.

### 3.6.3. Recommended Configuration

The amount of risk involved with a mission will dictate the level of security necessary. Defining this risk is outside the scope of this thesis. The Cryptography Library and security across all systems in general is recommended for all missions. At a minimum, authenticated encryption should be used on all critical spacecraft commands. Typically, on NASA funded missions, all received data is public record and must be shared at multiple levels of processing. Due to this, clear data can be used if desired on the downlink side, but security is always recommended. The amount of pre-loaded keys should be no less than what was utilized for interoperability testing. Due to the ability to upload new keys, the number of stored on-board keys may be reduced to meet mission requirements. Additionally, CRC tables could be replaced with calculations as needed to save memory, at the expense of processing required.

### 3.6.4. Debugging

For testing, additional debugging information is available. In the ‘crypto_config.h’ file, the debug defines are segmented out and clearly marked with (section)_DEBUG as the identifier. These debug messages utilize the print method provided by the OSAL to provide status and more detailed information apart from simply running behind the scenes, as the library would typically do during operations. For mission utilization it is recommended that errors and SDLS-EP log messages be recorded utilizing the cFS event services methods. Depending upon the current method currently under maintenance, only the desired debugging methods need be enabled. Table 7 provides details on the debug definition statements along with an explanation for each.
<table>
<thead>
<tr>
<th>Field</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARC_DEBUG</td>
<td>Anti-Replay Counter</td>
</tr>
<tr>
<td>CCSDS_DEBUG</td>
<td>Print out CCSDS messages prior to pushing onto the software message bus</td>
</tr>
<tr>
<td>DEBUG</td>
<td>Print out when entering and leaving the base external Cryptography Library functions</td>
</tr>
<tr>
<td>FECF_DEBUG</td>
<td>Frame error control</td>
</tr>
<tr>
<td>MAC_DEBUG</td>
<td>Message authentication code</td>
</tr>
<tr>
<td>OCF_DEBUG</td>
<td>Operational control field</td>
</tr>
<tr>
<td>PDU_DEBUG</td>
<td>Payload data unit</td>
</tr>
<tr>
<td>SA_DEBUG</td>
<td>Security association</td>
</tr>
<tr>
<td>TC_DEBUG</td>
<td>Telecommand</td>
</tr>
<tr>
<td>TM_DEBUG</td>
<td>Telemetry</td>
</tr>
</tbody>
</table>

Table 7 - Debug Fields

For official testing with the ESA, only a single debug field was active, the PDU_DEBUG, to enable capture of the processed data for the logs. During unofficial ESA testing each of these debug fields was utilized to verify the functionality it is associated with.
CHAPTER 4 | TESTING

4.1. Local Testing

For local testing a separate Python script is used to communicate directly with the FSW. This Python script is named “crypto_test.py” and allows for the sending of packets, from configuration files, to the FSW or the CI Lab application. This script simulates the output from the CI SLE Link script and enables the user to send the packets at a desired rate. Each packet is loaded sequentially and the name is shown on the screen. Once ready, the user may press ‘enter’ to send the packet and load the next one until the list of test packets is at an end. The FSW responds as if communicating in the standard configuration, and if this response is expected the script will be waiting for the response. An example of this script running side-by-side FSW is shown below in Figure 42. Returning TM frames must be examined manually in the present state.

Currently, scenarios exist to locally test both intra and intro-operability requirements as listed in the SDLS-EP protocol, like interacting with all services provided by the standard and error handling. This testing shows that the Cryptography Library goes beyond the initial expectations of simply supporting baseline mode. Modifying configurations and initial settings will results in failures and will require modification of the test cases. Each packet used for testing is stored individually in binary files.
Figure 42 - Cryptography Library Local Testing
4.2. Interoperability Testing

4.2.1. Objectives

The SDLS-EP testing documentation can be separated into two phases. The first phase is local testing or intra-operability testing performed solely by the ESA, while the second phase details interoperability testing between the ESA and NASA utilizing the Cryptography Library. Both phases have tests detailing the three services provides detailed in the SDLS-EP protocol in section 2.5.4. All testing is to be performed utilizing a bi-directional space link, with TC as the uplink and TM as the downlink, secured by both SDLS-CP and SDLS-EP baseline mode. The testing of error cases is limited to what can be detected utilizing the FSR, but more scenarios have been tested unofficially to get to this point.

This testing is to be fully documented and repeatable. Both modes, clear and secure, shall be tested. COP-1 operations and interactions are omitted for this due to integration with the CI Lab application and not the standard CI application, as used in the previous work. Both nominal and off-nominal cases shall be included, with both single and multiple errors occurring at once.

4.2.2. Configuration

The initial configuration is defined by the SDLS-EP testing procedures. This includes the mission-specific definitions utilized as well as the keys and specific values for each, along with the security associations. The summarized version of these is below:

- **General**
  - Encryption suite shall be AES-GCM as defined in
  - Key length shall be 128 bits
  - IV shall be 96 bits
  - Output MAC shall be 128 bits
  - All zeros (0) and all ones (65535) are reserved SPIs

- **Master Keys**
  - 0 - 000102030405060708090A0B0C0D0E0F -> ACTIVE
  - 1 - 10112131415161718191A1B1C1D1E1F -> ACTIVE
  - 2 - 202122232425262728292A2B2C2D2E2F -> ACTIVE

- **Session Keys**
  - 128 - 0123456789ABCDEF0123456789ABCDEF -> ACTIVE
  - 129 - ABCDEF0123456789ABCDEF0123456789 -> ACTIVE
  - 130 - FEDCBA9876543210FEDCBA9876543210 -> ACTIVE
  - 131 - 9876543210FEDCBA9876543210FEDCBA -> ACTIVE
4.2.3. Setup

In order to recreate the interoperability tests performed, the following systematic guide has been created. This guide was originally written to allow others to demonstrate the Cryptography Library for the CCSDS Security Working Group. The bullets below are to be followed in order, one by one, to produce Figure 43, Figure 44, Figure 45, Figure 46, and Figure 47.

- Sign in and boot server via Cloud Sigma’s website: https://zrh.cloudsigma.com/ui/#/
- Select the server configuration by clicking the link
- Enable VNC Tunnel and click “Display VNC” link once it appears
- Login: NASA / itc123!
- Open four individual terminals (Ctrl + Alt + T) and resize
  - Note that resizing terminals in the VNC is a hassle and should be avoided during operation.
- Terminal 1 – Core Flight System with Cryptography Library
  - To build:
    - cd ~/cFS_SDLS/
    - . ./setvars.sh
    - cd build/cpu1
    - make config && make
  - To run:
    - cd ~/cFS_SLDS/build/cpu1/exe
    - sudo ./core-linux
Figure 43 – Cryptography Library Demo Terminal 1

- Terminal 2 – Space Link Extension Application
  - To build:
    - cd sle_ivv/build
    - cmake .
    - make
  - To run:
    - cd sle_ivv/build/sle
    - sudo ./sts_sle
Figure 44 – Cryptography Library Demo Terminal 2

- Terminal 3 – Command Ingest Link (python)
  - To test with ESA:
    - cd cFS_SDLS/support/
    - python CI_SLE_Link.py
  - To test locally:
    - cd cFS_SDLS/support/test
    - python cFS_CI.py

Figure 45 – Cryptography Library Demo Terminal 3

- Terminal 4 – Telemetry Output Link (python)
  - To run:
    - cd cFS_SDLS/support
    - python TO_SLE_Link.py

Figure 46 – Cryptography Library Demo Terminal 4
4.3. Simulation To Flight 1 – Example Mission Implementation

STF-1 is West Virginia’s first spacecraft, as captured in the mission logo shown in Figure 48, and was constructed at NASA IV&V in collaboration with WVU. This mission, and many of the same caliber, do not consider security at any point during the lifetime. With the risk of failure already so high, the science data required to be released to the public, and the minimal amount of time and effort available, a project manager would agree that security is not a major concern. The simple fact is that all assets should be protected, not only to reduce risk to their own mission, but to reduce risk to other space missions and ensure that the money and time spent to achieve the goals of the mission are worthy.
The Cryptography Library had a requirement of straightforward deployment to a developed mission. The goal of this section is to provide a detailed example of how effortless implementing security on a spacecraft can be.

### 4.3.1. Interfacing with mission

The STF-1 mission utilizes an L3 Cadet half-duplex UHF Radio. The fine details of this radio are omitted to avoid potential regulatory issues, but proprietary protocols are involved for communications. This radio application communicates messages to the hardware, just as the TO Lab application forwarded telemetry out over UDP to the connectors during interoperability testing. The same implementation of sending the buffer of data off to be processed and have security applied could be used without higher-level protocols such as TC and TM. With the low risk STF-1 poses to other missions, telemetry could even be sent over a clear channel only using the Cryptography Library to preprocess commands received from the ground to perform operations.

Assuming the SDLS-EP protocol would be used in the TC side, integration into the ground data system would need to be performed. For STF-1, the open source COSMOS user interface for command and control from Ball Aerospace is utilized. This software, although written in Ruby, can have the Cryptography Library, written in C, wrapped and available for use to apply security to each command prior to transfer over the space link.
CHAPTER 5 | CONCLUSION

5.1. Summary

“Security is not simply created through agencies and operators; it is also achieved through coordination with manufacturers, software developers and operators. Numerous parties contribute to developing the integrated system for typical satellite operations, and as with any complex technical architecture, the more parties are involved, the greater the vulnerability. Further problems arise as space becomes more cluttered. For example, there is a severe lack of available frequencies for space based communications; orbit allocation is becoming increasingly problematic; and the amount of space junk is rising to critical levels.” [7]

The Cryptography Library is an answer to the issue of providing security to the software aspects of spacecraft systems in the near term. The ease of application and flexibility of both the library and the SDLS-EP protocol allow for use of a multitude of missions, and not just those that collaborate with NASA either due to the acceptance of the standard by the international CCSDS community. The application of the Library in the current form can occur on any platform with the ability to run C. Assuming the effort was too great to port the library from cFS, the library would still have the potential to be on manned missions, UAVs, rockets, and on all of the missions with a heritage with cFS.

This library has been designed and developed solely by John P. Lucas since the update to the Space Data Link Security Extended Procedures. Future updates have already been planned and are discussed in the next section. This work would not have been possible without the support of NASA IV&V and the JSTAR team.

5.2. Future Work

Development of additional functionality is still in progress for Cryptography Library. A few of these are elaborated upon in the following sub-sections:
5.2.1. Open Source Community

The Core Flight System is working toward expanding into an open source community, similar to what the RTEMs operating system developed and maintains. The benefits of this include feedback and contributions from a variety of people with varied use cases. Questions and answers could take place through a message board, in addition to a ticketing system to report bugs and provide a way to request new features. Modern configuration management systems, such as GitHub, provide many of these features. Internal to NASA, these systems already exist and are in use, but the community involvement is severely limited.

5.2.2. Encryption Algorithm Expansion

Upon presentation of the Cryptography Library at the 2017 Flight Software Workshop, many comments expressed concern for the outdated algorithms utilized by CCSDS. Luckily, the SDLS-EP planned for updates like this by providing a field to determine the algorithms used. Upon this release, the recommended standards from CCSDS were upheld. Roadmaps show plans to upgrade to AES-256 and increase key sizes.

5.2.3. Flight Hardware and FPGA Testing

Testing of the Cryptography Library on STF-1 flight hardware was planned, but due to an aggressive schedule, the flat-sat stage of integration did not provide ample time. If the library proved to be overly resource constraining for a particular mission, this functionality could be moved off-board between the radio and the main processor. An FPGA would serve this purpose well, and given that the library is already written in C, the effort would be minimal.

5.2.4. Deployment to the NASA Operational Simulator for Small Satellites

The STF-1 mission has been driving requirements in the NASA Operational Simulator for Small Satellites or NOS3, as seen in Figure 49. The primary goal of STF-1 is to develop and demonstrate the lifecycle value of a software-only small satellite simulator.
This simulator is open source and operating on an Ubuntu Linux virtual machine comprised of multiple components to enable the running of FSW with simulated hardware components and a ground station to allow for operational or day in the life testing of the spacecraft prior to launch. The Cryptography Library has been approved for inclusion in the sample application set to be deployed with NOS³ in the next release to further demonstrate the ease of application to a generic mission implementation and promote security.

5.3. Remaining Questions

A number of questions were raised during the progression of this work. At the finalization of the paper, a few still remain to be determined by the CCSDS Security Working Group. Questions of the standard that go unanswered:

- **GVCID**
  - No standard has been detailed for the use of GVCIDs in the SDLS-EP protocol. In the current form of the Cryptography Library, GVCID is defined as four bytes. The first byte represents the type: TC, TM, or MAP. The three remaining bytes designate the VCID.

- **Master key vs session key determination**
  - No ratio of master to session keys is ever discussed. In the current implementation all keys are stored in the same key ring, or array in memory. Keys with a value less than 128 are designated master keys. This value is hard coded, but could easily be changed dependent upon mission configuration.

- **Packet Utilization Standard**
This standard was elaborated upon in Section 2.7.6.3 and is included in the Cryptography Library. Due to the fact that NASA has not adopted this standard, support is limited in the library. For NASA missions it is recommended that this be removed for simplicity.
CHAPTER 6 | BIBLIOGRAPHY


5.4. Open Source Paperwork

Global Concerns Statement

The Global Concerns Statement is prepared by asking the developer of the software to provide a statement reflecting on whether the subject software:

- Could be used in any way that would be contrary to the best wishes of Goddard, NASA, or the U.S. of America

How could your software be used in this manner? If not, please state why?

No, it could not.

- Might divulge trade secrets or risk giving away any "high technology"

How might your software be used in this manner? If not, please state why?

No, it would not.

- Contains source code that could be modified to do harm (command a spacecraft or disrupt Wall Street)

Does your software contain code or can be used in this manner? If not, please state why?

No, would need encryption keys used by satellite.

- Might harm other software or operational systems

How might your software do harm in this manner? If not, please state why?

No.

- Utilizes encryption

Does your software utilize encryption and if not, please state why?

Yes.
GSFC SOFTWARE PUBLIC DISCLOSURE
EXPORT CONTROL CHECKLIST

Descriptive Text
Core Flight System Cryptography Library

Core
The NASA Export Control Program requires that the external release of GSFC software be reviewed for export control considerations in accordance with the Export Administration Regulations (EAR 15CFR 730-774), the International Traffic in Arms Regulations (ITAR 22CFR 120-130), and NASA Policy Directive 2210.1 along with NASA Procedures and Guidelines 2210.1 both entitled "External Release of NASA Software." Release of NASA software into a public forum may provide countries with interests adverse to the U.S. with access to NASA technology. Failure to comply with the ITAR and/or the EAR may subject you to fines of up to $1 million and/or up to ten years imprisonment per violation. This checklist cannot be used if the dissemination of software is limited in any manner, such as through distribution to specific companies or organizations as a result of contracts, or licensing agreements. The following process cannot be utilized for the public release of source code or encrypted software.

Generally, the export of software pertaining to the design, development, production, manufacture, assembly, operation, repair, testing, maintenance or modification of defense articles, i.e., space flight hardware, ground tracking systems, launch vehicles to include sounding rockets and meteorological rockets, radiation hardened hardware and associated hardware and engineering units for these items are controlled by the State Department under the ITAR. A complete listing of items covered by the ITAR can be accessed at http://bmd.gsfc.nasa.gov/export/ejssitars.htm. The export of software with respect to ground based sensors, detectors, high-speed computers, and national security and missile technology items are controlled by the U.S. Commerce Department under EAR. If the software intended for release falls within the above categories but fits into one or more of the following exemptions, the software may be released.

EXEMPTION I
If your software is already in the public domain in its entirety through a non NASA medium and/or through NASA release previously approved by the Export Control Office, the information is exempt from further review. If the information falls into this category, you may attest that you are using this exemption by signing below.

Name and Code (type or print)  Signature and Date

EXEMPTION II
If your software is utilized for the development or display of scientific data, i.e., data pertaining to studies of clouds, soil, vegetation, oceans, and planets, without the disclosure of information pertaining to articles controlled by the ITAR or EAR, such as flight instruments, high speed computers, or launch vehicles, the software is exempt from further review. If the software falls within this category, you may attest that you are using this exemption by signing below.

Name and Code (type or print)  Signature and Date

EXEMPTION III
If the software does not fall within the two exemptions listed above, software may be released using ITAR exemption 125.40(b)(3). Use of this exemption is afforded only to agencies of the Federal Government and allows for the release of ITAR controlled software into the public domain. The GSFC Export Control Office has determined that use of this exemption will be allowed only after receiving assurance that such release is a responsible action. The following internal guideline has been established regarding the use of this authority: That the software does not offer specific insight into the design, development, production, or manufacture of an identified ITAR controlled item (ref. Para. 3) in sufficient detail (by itself or in conjunction with other tech data) to allow potential adversaries to replicate the item, or exploit/delcast controlled U.S. technologies. If you determine that you have met this criteria, you may attest your determination by signing below, and the GSFC Export Control Office will offer favorable consideration.

Name and Code (type or print)  Signature and Date

John P. Lucas, Code 180
/* Copyright (C) 2009 - 2017 National Aeronautics and Space Administration.
All Foreign Rights are Reserved to the U.S. Government.
This software is provided "as is" without any warranty of any, kind either
express, implied, or statutory, including, but not limited to, any warranty
that the software will conform to, specifications any implied warranties of
merchantability, fitness for a particular purpose, and freedom from
infringement, and any warranty that the documentation will conform to the
program, or any warranty that the software will be error free.

In no event shall NASA be liable for any damages, including, but not limited
to direct, indirect, special or consequential damages, arising out of,
resulting from, or in any way connected with the software or its
documentation. Whether or not based upon warranty, contract, tort or
otherwise, and whether or not loss was sustained from, or arose out of the
results of, or use of, the software, documentation or services provided hereunder

ITC Team
NASA IV&V
ivv-itc@lists.nasa.gov
*/
5.5. Presentation

Core Flight System Cryptography Library

John Lucas
John.P.Lucas@nasa.gov

www.ccsds.org
Overview

- Why do we need security?
- Security Incidents
- Background
  - cFS
  - CCSDS
- Previous Work
- Cryptography Library Design
- Testing
- Demonstration
- Example Mission: STF-1
- Future Work
Why do we need security?

- Obscurity and uniqueness are not sufficient protection
  - The technology for spacecraft communication is more readily available
  - Use of public ground networks to transfer mission control and monitoring data has increased vulnerability

- Spacecraft systems are a high-profile target
  - An attacker can cause considerable harm to a nation or company by compromising a spacecraft
Security Incidents

• Chaos Communication Camp 2015
  – Rad10 Badges
  – “The problem isn’t that Iridium has poor security. It’s that is has no security.”

• NASA / NOAA
  – At least two US environment-monitoring satellites have suffered interference four or more times in 2007 and 2008.
  – Satellite Data Information System was taken offline in September 2014 after hacking incident.

• Turla Advanced Persistent Threat
  – Russian cyber-espionage group exploiting older satellites that do not support encrypted connections.
Motivation

- Adversaries
- Commercialization of space
- Cost of access
- Increased number of satellites
  - Small, Micro, Nano, and Pico
  - Agencies, Schools, and Hobbyist
- Mission complexity
- Standardization
  - Constellations or Swarms
- Space agency benefits
Background
Core Flight System

- Open source flight software developed by GSFC
- Operating system abstraction layer
- Platform support package
Consultative Committee for Space Data Systems

- Founded in 1982 by the major space agencies of the world
- Experts from 26 nations collaborate
- Primary goal is interoperability between space agencies
- More than 800 space missions flown

- cFS was developed with these protocols in mind:
  - Space Packet Protocol (SPP)
  - CCSDS File Delivery Protocol (CFDP)
  - Communications Operation Procedure – 1 (COP-1)
  - Telecommand Space Data Link Protocol (TC)
  - Telemetry Space Data Link Protocol (TM)

- Goal to evolve with Space Data Link Security (SDLS)
Protocol Layers

- Space Link Extension
  - Forward Communications Link Transmission Unit (FCLTU)
- Telecommand (TC)
- Space Packet Protocol (SPP)
- Space Data Link Security (SDLS)
- Telemetry (TM)
- Space Link Extension
  - Return Channel Frame (RCF)
Space Data Link Security

- Compatible with both the TC and TM protocols
  - Those used with the CI and TO applications
- Structured method for applying data authentication and/or data confidentiality at the Data Link Layer
- Applies a security header and trailer to the payload

<table>
<thead>
<tr>
<th>Space Link Protocol Frame Headers (may be multiple)</th>
<th>Frame Data</th>
<th>Space Link Protocol Frame Trailers (may be multiple)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Security Header</th>
<th>Frame Data (secured by authentication and/or encryption)</th>
<th>Security Trailer (optional)</th>
<th>Space Link Protocol Frame Trailers (may be multiple)</th>
</tr>
</thead>
</table>
## Space Data Link Security

- **TC:**

```
<table>
<thead>
<tr>
<th>TC TRANSFER FRAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANSFER FRAME PRIMARY HEADER</td>
</tr>
<tr>
<td>(bits) 5</td>
</tr>
</tbody>
</table>
```

- Encryption is done on Frame Data only.
- MAC is computed over Primary Header, Segment Header, Security Header, and Frame Data.
- MAC is not computed over CIU Start Sequence (not shown) or ECP.
- TC Apply Security Payload

- **TM:**

```
<table>
<thead>
<tr>
<th>TM TRANSFER FRAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANSFER FRAME PRIMARY HEADER</td>
</tr>
<tr>
<td>(bits) 6</td>
</tr>
</tbody>
</table>
```

- Encryption is done on Frame Data only.
- MAC is computed over Primary Header, Secondary Header, Security Header, and Frame Data.
- MAC is not computed over ASM (not shown), ODF, or EDF.
- TM Apply Security Payload

---

4/18/2017

NASA IV6V - JSTAR - NON-JSTAR
APPROVED FOR PUBLIC RELEASE
Space Data Link Security

- Security Header:

- Security Trailer:
Previous Work

- Interoperability testing with CNES
  - CCSDS Security Working Group
- Required prior to publishing of SDLS Core Protocol
  - AOS protocol on downlink side tested
- Utilized GPM Operational Simulator (GO-SIM)
  - cFS and ASIST ground station
- Simple Cryptography Library
  - Advanced Encryption Standard (AES) algorithm
    - Galois Counter Mode (GCM-AES)
  - Cipher-Based Message Authentication Code (CMAC)
  - Frame decoding/encoding performed in CI and TO Applications
Previous Work
Previous Work

AOS-SDLS Implementation

Generate Secure Frames

E-Mail Secure Frames

Process Secure Frames

Compare

Original Frames

Processed Frames
Space Data Link Security – Extended Procedures

- Extended Procedures provides three services:
  1. Key Management
     - Ensures both communications endpoints are synchronized in terms of cryptographic keys and key states.
     - Over-The-Air-Rekeying can be used to update session keys.
  2. Security Association (SA) Management
     - Provides a means to setup, activate, check status, and control an SA of the remote system.
     - As missions become more complex, dynamic management will be necessary.
  3. SDLS Monitoring and Control
     - Ability to modify mission specific parameters
     - Maintains the status of the security processor as a whole.
Space Data Link Security - Extended Procedures

1. Key Management
   - Symmetric key management infrastructure
   - Scalable to meet mission needs
   - Master keys
     - Used to decryption session keys
   - Session keys
     - Used for all other message types
   - Over-The-Air-Rekeying (OTAR)
2. Security Association (SA) Management
   - Anti-replay counter / window size
   - Authentication bit mask
   - Direction – TC or TM
   - Encryption cipher suite
   - Linked keys
   - Managed lengths of values
Space Data Link Security – Extended Procedures

3. SDLS Monitoring and Control

- Commands
  - Ping
  - Manage Logs
  - Read Sequence Number
  - Reset Alarm
  - Self-Test
- Logs and Reports
  - Frame Security Report (FSR)

---

Frame Security Report Table:

<table>
<thead>
<tr>
<th>CONTROL WORD TYPE &lt;&lt;1&gt;&gt;</th>
<th>FSR VERSION NUMBER &lt;&lt;100&gt;&gt;</th>
<th>ALARM FIELD</th>
<th>SECURITY EVENT FLAGS</th>
<th>LAST SPI USED</th>
<th>SN VALUE (LSB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(bits)</td>
<td></td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>16</td>
</tr>
</tbody>
</table>

---

4/18/2017

NASA JPL - JSTAR – NON-JSTAR
APPROVED FOR PUBLIC RELEASE
Requirements

• Cross-platform codebase
  – Can be used by flight software and the ground system
  – Developed in C

• Minimal to provide only necessary functionality
  – AES 128-bit block cipher
  – 128-bit Galois/Counter Mode for Authenticated Encryption
  – 128-bit CMAC authentication

• Open-Source
  – Currently pending approval

• Simple to implement
  – ApplySecurity(ingest, len_ingest)
  – ProcessSecurity(ingest, len_ingest)
Architecture
Architecture

- ESA required SLE communications
- Links to cFS created in python
- CI Lab Application
  - Receives TC frames via UDP
  - Calls Process Security
- Cryptography Library
  - SDLs-EP commands interpreted and response generated if required
- Additional Spacecraft Processing
- TO Lab Application
  - Receives SPP frame and packages into TM via Apply Security
Demo
Example Mission: Simulation To Flight – 1

- Small Satellite – 3U
- One simplex Cadet UHF radio
  - Store and Forward
- Radio Application
  - Supporting SPP and proprietary radio communication packets
  - Simply call Apply and Process Security
- COSMOS Ground Station
  - Configure to process TC and TM protocols
  - Embed Cryptography Library
Future Work

- Additional Encryption Cipher Types
- Advanced Orbiting Systems (AOS) Support
- Ground System Integration
  - Utilizing COSMOS
- Load Testing
  - Implementation on AVR32 flight computer
- Improved separation / explanation and support of mission specific parameters
- Inclusion into simulator platforms
  - NASA Operational Simulator for Small Satellites (NOS³)
- Integration into mission
- Additional Unit Tests
Conclusion

- Continued support of CCSDS from NASA
- Effects on future missions

- Presentations
  - 2016 Flight Software Workshop
  - 2017 Small Satellite Conference (Future)

- Protocol Updates
  - Security Working Group Discussions

- Publications
  - CCSDS Cloud Testing
  - Space Data Link Security Extended Procedures
Questions?

John Lucas
John.P.Lucas@nasa.gov

www.ccsds.org
This page intentionally left blank.