**NTNU – Trondheim**
Norwegian University of
Science and Technology

# Access control of NUTS uplink

Sandesh Prasai

Master of Telematics - Communication Networks and Networked Services (2
Submission date: July 2012
Supervisor: Stig Frode Mjølsnes, ITEM
Co-supervisor: Roger Birkeland, IET

Norwegian University of Science and Technology
Department of Telematics

# PROBLEM DESCRIPTION

Name of student: Sandesh Prasai
Course: TTM4905, Master Thesis
Thesis title: Access control of NUTS uplink

**Thesis description:**
The radio link protocol of the NUTS satellite will probably be (some version of) AX.25. This protocol does not give us any means of authenticate commands. It has been proposed to use CubeSat Space Protocol (CSP) to transfer data between internal modules in the satellite, as well as between the satellite and the ground station. The student should study previous analysis work on the radio link access control and CSP, and then propose a design and an implementation of the CSP for the satellite microcontroller.

In addition, security in a proposed network of ground stations (GENSO) could be addressed and discussed.

Department: Department of Telematics
Supervisor: Roger Birkeland
Responsible Professor: Stig Frode Mjølsnes
Date: 04/07/2012

# ABSTRACT

Access control of NUTS uplink

The goal of this thesis is to investigate the cryptographic parameters in access control of NTNU Test Satellite (NUTS), and to make recommendations for secure uplink of NUTS. NUTS is a nano-satellite popularly known as CubeSat which is being developed by NTNU. The ease and low cost in construction cost shows that there is growing demand for developing and launching CubeSat. This thesis analyzed the communication protocol, security protocol, protection against replay and integrity of uplink commands for NUTS.

The brief introduction of NUTS and requirement of security parameters were presented along with the scope of thesis. The analysis of the security protocols used by similar CubeSat projects was done, nonetheless very few had implemented a security protocols. The security issues in wireless communication systems were discussed with the advantages and disadvantages of different security protocols. Also, NUTS hardware and software platform was discussed to insight the NUTS architecture.

The integrity protection was focused than the confidentiality of message in uplink. Therefore, this thesis analyzed the threats in protocol adopted for communication between internal modules of satellite as well as between space and ground station. In addition, the possible ways of mitigation were explained. Here, three different hash functions were tested in AVR platform to check the computation efficiency in microcontroller. The efficient hash was proposed to use in HMAC for integrity protection. The different key exchange mechanisms were also discussed to address the key distribution and resynchronization between satellite and ground segment.

Finally, some modifications were recommended in existing protocol that makes the system secure against the integrity and replay. The results also show that the addition of sequence number to check the freshness of message, HMAC code and CRC is feasible to transmit within AX.25 frame. Last but not the least, further modification to this research and possible future extension were proposed.

# ACKNOWLEDGMENT

First and foremost, I would like to thank my department for giving me an opportunity to work with *Access control of NUTS uplink* and department of electronics for offering this thesis topic. I am pleased to express my sincere gratitude towards Professor Stig Frode Mjølsnes at department of Telematics, Norwegian University of Science and Technology (NTNU), for his excellent guidance and readiness to help during my work. His valuable feedback and suggestions throughout the thesis and report writing phase were incredible. Further, I would like to pour sincere thanks to my co-supervisor Roger Birkeland for his immense help during entire thesis phase and providing valuable input in my work.

I wish to thank Professor Danilo Gligoroski for providing the source code of hash functions BMW and Edon. His guidance in interpreting result was very helpful for concluding output. Last but not the least, I would like to thank my friends and may family who helped me directly and indirectly to accomplish this thesis.

Sandesh Prasai
Trondheim, Norway,

# PREFACE

This text is submitted as partial fulfillment of the requirements of degree in MSc in Telematics at the Norwegian University of Science and Technology (NTNU). This master thesis has been carried out at the Department of Telematics under the co-operation with Department of Electronics (NTNU) during the period February from 2012 to July 2012 under the guidance and supervision of Professor Stig Frode Mjølsnes (Telematics department) and Mr. Roger Birkeland (Project manager of NUTS CubeSat). I would like to thank both of them for their immense guidance and feedback throughout my work. Their comments have helped me a lot to accomplish this thesis with fruitful outcome.

Trondheim, Norway, July 2012

_____

Sandesh Prasai

# TABLE OF CONTENTS

## Contents

# LIST OF FIGURES

# LIST OF TABLES



‘

# ACRONYMS

| | |
|---|---|
| NUTS | NTNU Test Satellite |
| LEO | Lower Earth Orbit |
| CalPoly | California Polytechnic State University |
| SSDL | Space Systems Development Laboratory |
| NAROM | Norwegian Centre for Space-related Education |
| OBC | ON Board Computer |
| ADCS | Attitude Determination and Control System |
| AES | Advance Encryption Standard |
| SHA-1 | Secure Hash Algorithm-1 |
| BMW | Blue Mid-night Wish |
| MAC | Message Authentication Code |
| HMAC | Hash-based Message Authentication Code |
| CSP | CubeSat Space Protocol |
| XTEA | Extensible Tiny Encryption Algorithm |
| CRC | Cyclic Redundancy Check |

| | |
|---|---|
| TTP | Trusted Third Party |
| DoS | Denial of Service |
| OTP | One Time Password |
| NIST | National Institute of Standards and Technology |
| SUPERCOP | System for Unified Performance Evaluation Related to Cryptographic Operations and Primitives |
| TCP | Transmission Control Protocol |
| USB | Universal Serial Bus |
| GENSO | Global Education Network for Satellite Operations |
| AUS | Authentication Server |
| MCC | Mission Control Client |
| GSS | Ground Station Server |

# CHAPTER 1

## *INTRODUCTION*

This master thesis is carried out on the base of project report done by master student Vilius Visockas on date 2011-2012 [17]. Since, both the works are based on the same platform; it also possesses similar concepts of uplink security. However, this report is concentrated in the authentication and integrity in NTNU Test Satellite (NUTS) uplink rather than encryption.

Absence of security parameters in a payload results unauthorized user (hacker/attacker/malicious user) to retransmit the command or tamper the payload. They can intentionally make CubeSat to execute a false command and control upon the resources or even destroy the CubeSat. This may leads to denial of service from satellite or even loss of the satellite permanently. Therefore, it is necessary to protect our system to get access from such attacker. There are several cryptographic approaches to protect against the attacker. Here, we preferred to protect integrity of source and data rather than encryption. Encryption only protects the confidentiality of message but fails to protect integrity. Meaning that, it has no confidence from where the message is coming and the originality of data. In this scenario, it is very important to protect the integrity of message. In NUTS, we decided to use HMAC based integrity protection. NUTS use CubeSat Space Protocol (CSP) to communicate between the internal module as well as space and ground segment. This CSP includes SHA-1(secure hash algorithm-1) for integrity protection but we also checked the feasibility of other two different hash functions. In addition, we solved our major concern of replay attack with the implementation of sequence number (SN) in every CSP packet. We also analyzed the feasibility of SN and HMAC code in CSP and radio packets. The analyses of different key exchange mechanisms are also presented which we found infeasible to use trusted third party. Finally, the design of uplink transmitter from the ground segment and receiver of satellite segment were also presented. All these work were done within the limitation of CubeSat constraints and making system as simple as possible which was the great challenge.

This thesis has five chapters in which first chapter introduces NUTS briefly and the scope of this thesis. The objective and methodology are explained in this chapter which also includes the introduction of similar successful projects. The hardware and software platform used in NUTS project are explained in chapter two. Later part of this chapter points out the need of security in satellite communication systems.

The detail security analysis of cryptographic algorithms used in uplink is analyzed in chapter three. It also includes the analysis and necessity of secure hash function and key exchange mechanism. In chapter four, the best possible mechanisms and techniques are listed and explained which mitigates the security threats. The design of uplink and the downlink is also proposed in this chapter. Finally, chapter five consists of conclusion and the possible extensions of this work in future.

## 1.1 Scope of thesis

In every communication systems, the information should be protected in both the ways (i.e. uplink and downlink). However, in case of NUTS, the uplink data is only protected whereas downlink data is not encrypted at all. It contains educational data like weather information, pictures and videos of space which can be accessed by tuning a correct frequency. Thus, it is not vulnerable to access the downlink data from any other stations. Nevertheless, the uplink command must be sent by legitimate user. The task of employing a successful access control solution lies in determining the appropriate authentication and authorization mechanisms. On Board Computer (OBC) in CubeSat should verify the command or data received is being transmitted from the legitimate ground station. In this thesis, an uplink security of CubeSat is analyzed and the changes in CubeSat Space Protocol (CSP) are proposed. For secure uplink, the encryption algorithm is untouched whereas authentication and integrity is explained in detail.

This thesis work is carried out within the following objectives;

- To analyze the current CSP protocol and recommend the best approach for NUTS access control and uplink security.

- To substitute the hash function SHA-1 with either Blue Midnight Wish (BMW) or Edon-R in CSP for better security and efficiency in execution.
- To find a solution to the replay protection by using some counter mechanism
- To address the problem of resynchronization.

Current CSP protocol uses SHA-1 in HMAC and XTEA as encryption algorithm. Still, they cannot guarantee against the replay. The idea behind the substitution of message digest function SHA-1 by BMW or Edon-R is to use a different hash function which is efficient in terms of speed and size of code with higher security. To avoid more power consumption in satellite which is power limited, an efficient but highly secure hash function is proposed. The detail explanation is presented in chapter three about the choices of hash function. Hash function protects the integrity of the message which is also explained in chapter two. Resynchronization is an important issue in long distance wireless propagation. Attenuation and degradation of signals by interference make the signal unreachable in the destination. Similarly, delay in the propagation should be considered that breaks the synchronization in communication between source and destination. These effects are considered in the later chapter and the best approach for security and synchronization are proposed for NUTS.

## 1.2 Methodology

To test the computational speed and the size of hash function, we compiled all the three hash functions in Atmel AVR Studio 5.1 and Microsoft Visual Studio 11. We used visual studio to cross check the output of HMAC with AVR studio. After successful compilation of the code in Atmel AVR Studio 5.1 which was written in C, a hex code is generated. We then checked the size of generated hex file of each hash algorithm to determine the size content by it. This is the size of program code that resides in the memory of microcontroller. The reason we are focused towards the memory of microcontroller is due to limitation in its internal memory. AT32UC3A3256 which has 256 Kbytes is only memory which is responsible for processing and storing its software [11]. Hence, it is very important to consider the size of code during the choice of hash function. However, NUTS is working to expand external memory up to 16 MB which will eliminate the memory limitation problem. The computational speed is determined

simply by calculating the clock pulse difference before and after the executing source code. In other words, hashing the messages of different size and taking the average clock pulse required to digest the message. Moreover, Atmel AVR Studio 5.1 is also used to benchmark the execution speed.

Finally, different mechanisms were analyzed against the replay protection and key exchange mechanism. The most efficient key exchange mechanism, integrity protection and replay protection approach for the NUTS CubeSat was discussed and proposed.

## 1.3 Similar Projects

It is found that the CubeSats launched up to this date has not been focused intensely in its security. Most of the CubeSats uses AX.25 protocol and cyclic redundancy check (CRC) for radio packet transmission and error detection respectively. AX.25 is a link layer protocol which is used to transmit radio packets among the sender and receiver. Data is encapsulated in a frame and transmitted as a packet [26], the maximum packet size transmitted is 256 bytes. It provides very low level of security through bits in control field with checking the frame order, sequence and the arrival timing. Furthermore, CRC (Cyclic Redundancy Check) is used as an error detecting code in communication systems. The algorithm which is based on cyclic codes will generate a sequence of bits. These bits are appended with the original data during the transmission. Similarly, at the receiving end the CRC is recalculated and compared with the received one. It keeps the integrity of the message up to some level. However, it is not sufficient to protect against an intentional attacker. It is possible for an attacker to alter the message, recalculate the CRC value and retransmit towards the destination.

**SwissCube**

It is a Swiss CubeSat designed to conduct research into nightglow within Earth's atmosphere [28]. It uses AX.25 protocol to frame format in uplink and downlink as well. It used CRC with polynomial $x^{16} + x^{12} + x^5 + 1$ to check the error in payload [28].

| Flag | AX.25 Transfer Frame Header (128 bits) | | | | Information Field | Frame-Check Sequence | Flag |
|---|---|---|---|---|---|---|---|
| | Destination Address | Source Address | Control Bits | Protocol Identifier | | | |
| 8 | 56 | 56 | 8 | 8 | 32-2048 | 16 | 8 |

Figure 1: SwissCube AX.25 frame format, taken from [28]

In figure 8, frame sequence check value of 16 bit is assigned for CRC. But it has not implemented any additional security measures against integrity protection and replay.

**AUSAT II**

This CubeSat is built and operated by students from Alborg University in Denmark. It uses modified AX.25 protocol and AAUSAT-II protocol which was developed by them [30]. AAUSAT –II protocol contains the features of acknowledgement from satellite to earth station after the reception of data frames. Reception of NACK enables earth segment to resend data frame [31]. There is no security mechanism implemented besides error checking with ACK and NACK in AUSAT-II.

**AubieSat-1**

AubieSat-1 is the first student built CubeSat in Auburn University with volume and weight of 1000 cm3 and 1.03-kg respectively. This CubeSat studies radio wave propagation through the ionosphere test solar panel protective films [32]. Besides Morse code, it does not have any cryptographic approaches implemented to protect against the attacker or malicious user [33].

## 1.4 Summary

The outcome of this research work is expected to decide the implementation of security features in uplink of NUTS. Security parameters used in CubeSat Space Protocols are analyzed and some new changes are proposed. It considers the sequence number to protect against replay

and HMAC code to protect integrity. It was found that resynchronization of sequence number was an issue in long distance wireless propagation and this could be address by implementation of time stamp. Therefore, time stamping can be a better solution than sequence number.

# CHAPTER 2

## *THEORY BEHIND CUBESAT AND SECURITY*

### 2.1 Introduction to CubeSat:

CubeSat is a miniaturized satellite launched into a low-earth orbit (LEO) and used for education and scientific purpose. The idea of launching small satellite named CubeSat was first presented by California Polytechnic State University (CalPoly) and the Space Systems Development Laboratory (SSDL) at Stanford University. They targeted University students to provide knowledge about space science exploration [2]. CubeSats are constructed within many constraints which are shown in figure 2 [1].

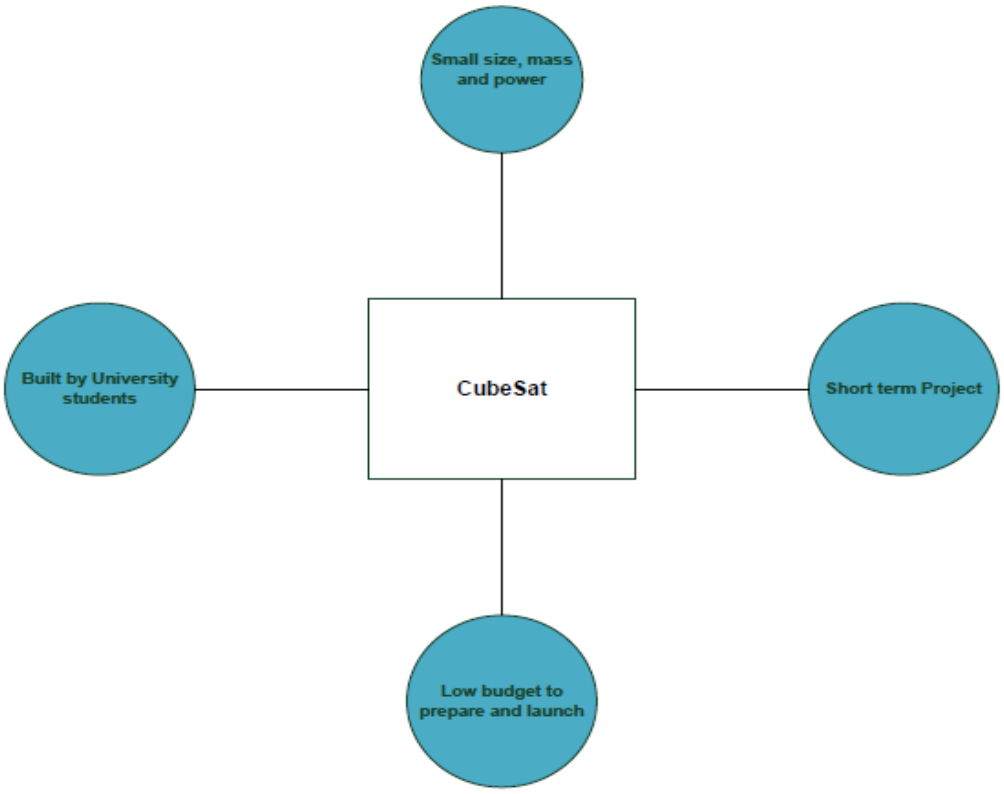


Figure 2: CubeSat Goals

After the successful launch of their CubeSat, several other Universities are following this trail by developing and deploying this sort of nano-satellite for space exploration and educational purposes.

The motivations behind popularity of CubeSat are as follows [3];

- The advancement of technology has greatly decreased the electronic chip size resulting in the production of complex and very small integrated circuits (ICs).
- Enormously increasing in the processing capability and reduce the power consumption.
- Student from the different faculties are involved in making CubeSat.
- It is cost friendly.
- Lower earth orbit allows higher data rates with less error because of short communication distance.

## 2.2 NUTS background

NTNU Test Satellite (NUTS) is the part of the Norwegian Student Satellite program which is hold by Norwegian Centre for Space-related Education [NAROM]. The goal of this project is to design, manufacture and launch a double CubeSat by 2014 [4]. It project is being carried out by students from various departments like; IET (Department of Electronics and Telecommunication), ITK (Department of Engineering Cybernetics), IPM (Department of Engineering Design and Materials), PHYS(Department of Physics), IDI (Department of Computer and Information Science) and ITEM (Department of Telematics). The project is run and managed by IET but students from different departments have their own Professor from respective department as a supervisor. The satellite lab and ground station room are located in the fifth floor which serves common place for all project members to work together, study, weekly progress meeting etc. Each student is working on their own respective area. These areas are categorized as communication, internal data bus OBC and power management, ACDs, payload, mechanics and system. Including NUTS, CUBEstar at university of Oslo and

HINCube at Narvik University College are two other CubeSat projects run by NAROM. However, NUTS have its own model including size (10*10*20) cm, communication bus, security protocols etc.

## 2.3 Implementation Platform

### 2.3.1 Microcontroller AVR32UC3

AVR32UC3 is a 32-bit RISC microprocessor architecture designed by Atmel. It was designed by Øyvind Strøm and Erik Renno in Atmel's Norwegian design center. In this microcontroller, most instructions are executed in single-cycle. It is the most efficient microcontroller with high performance and low power consumption [5].
Some of the key features of AVR UC3 are listed below [6];

- Software Library — All UC3 devices are supported by AVR Software Framework, a complete library of device drivers and middleware.
- Connectivity — USB device and host, Ethernet MAC, SDRAM, NAND flash, and fast serial interfaces are ideal for complex applications.
- Arithmetic performance — Integrated FPU increases precision and dynamic range in digital signal processing.
- High data throughput — Peripheral DMA, multi-layered databus and large on-chip SRAM remove bottlenecks in high-speed communication.
- Low power consumption — picoPower® technology delivers the industry's lowest power consumption in active and sleep modes.

In NUTS project, we are using AT32UC3A3256 microcontroller which has inbuilt software library. Out of all, encryption library is the one function that supports cryptographic functions for adding security to any application [6]. The library contains AES, 3DES, MD2, MD4, MD5, SHA1, SHA256, RSA1024, X.509 TLS version 1 and SSL version 3. These cryptographic

functions can be used for secure communication between parties. These are also integrated with other libraries like TCP/IP and USB communication stack.

### 2.3.2 FreeRTOS

FreeRTOS is a very small portable, robust and open source operating system. Therefore, it is suitable for small applications on small platforms. Due to this feature many processors and microcontrollers implement FreeRTOS. Some features of FreeRTOS that make it a most popular operating system as explained in [7]are listed below.

- Very small memory footprint, low overhead, and very fast execution.
- Scheduler can be configured for both preemptive and cooperative operation.
- Co-routine support (Co-routine in FreeRTOS is a very simple and lightweight task that has very limited use of stack)
- Trace support through generic trace macros. Tools such as FreeRTOS+ trace (provided by the FreeRTOS partner Percepio) can thereby record and visualize the runtime behavior of FreeRTOS-based systems. It includes task scheduling and kernel calls for semaphore and queue operations.

The OBC in NUTS project uses FreeRTOS because CSP we using is ported to FreeRTOS.

### 2.3.3 AVR Studio 5.1

This toolkit is the Integrated Development Environment (IDE) for developing and debugging embedded Atmel AVR applications. It is simple to use and supports all 8-bit and 32-bit AVR microcontrollers. The AVR Studio® 5.1 editor simplifies code editing and let us to write code more efficiently. This IDE connects easily to Atmel AVR debuggers and development kits for C/C++ and assembler code [10].

## 2.4 Ground Segment equipment

In order to communicate with space segment, various software and hardware equipment are required in ground segments. Those are mentioned as follows;

- Computer with Window XP operating system.
- Antenna Yagi-Uda for UHF and VHF.
- WXTrack and decoders for satellite tracking.

## 2.5 Security Issue

Here, some security related issues during communication between ground station and satellite station are discussed. Similarly, simple principle of some cryptographic mechanism like encryption, MAC and HMAC are also mentioned. However, this thesis is focused solely in integrity.

**Risk with unsecure satellite communication**

There are, of course, risks to unsecured satellite based communication. Specifically, if one got access to a satellite systems then that may damage the satellite or even disastrous consequences may ensue. What might happen, for example, if sensitive military information is disclosed? What if the upcoming information of natural disaster is prevented to transmit? What if high level diplomatic communication is intercepted and replied? All these may lead to the unwanted consequences. The likelihood of attack is low in educational satellite like CubeSat which is also mentioned in project report by Vilius [17]. However, author in [25] describes gaming, profit or revenge and ego as the source of the common motivations for an attack are popular in wireless networks. According to [25], a hacker who enjoys his free time in tracking and attacking other's system is gaming attack. Attackers are also motivated with profit or revenge, such that collapsing and corrupting of other's system may lead to the profit or tangible reward to an attacker.

**Need of encryption**

The area covered by the satellite beam on the earth surface is called footprint. The signal transmitted by the satellite disperses in conical fashion as it approaches the surface of the earth. Thus, signal may be made available over a wider geographic area than would be optimally desirable. In the absence of encryption, anyone tuning the radio in that region can receive the message. Confidentiality protects against the malicious disclosure of information but in CubeSat only the uplink data are encrypted. The uplinked payload is encrypted in CubeSat so that attacker without key cannot read the data easily. Generally, simple encryption algorithm is preferred in CubeSat due to its design. The downlink data is kept unencrypted so that anyone can tune the correct frequency and extract the information. This is because; it contains only educational information about space behavior which does not need much that of security like that of Defense System.

**Symmetric Encryption**

Also known as private key encryption in which same key is used for both encrypting and decrypting of message. For instance, if party A wants to communicate with party B via satellite using symmetric encryption method, then both A and B must agree on a unique secret key. As long as the key remains secret, it serves to authenticate both parties. Advance Encryption Standard (AES), RC4, Data Encryption Standard (DES) and Triple DES or 3DES are the commonly used symmetric encryption algorithms.

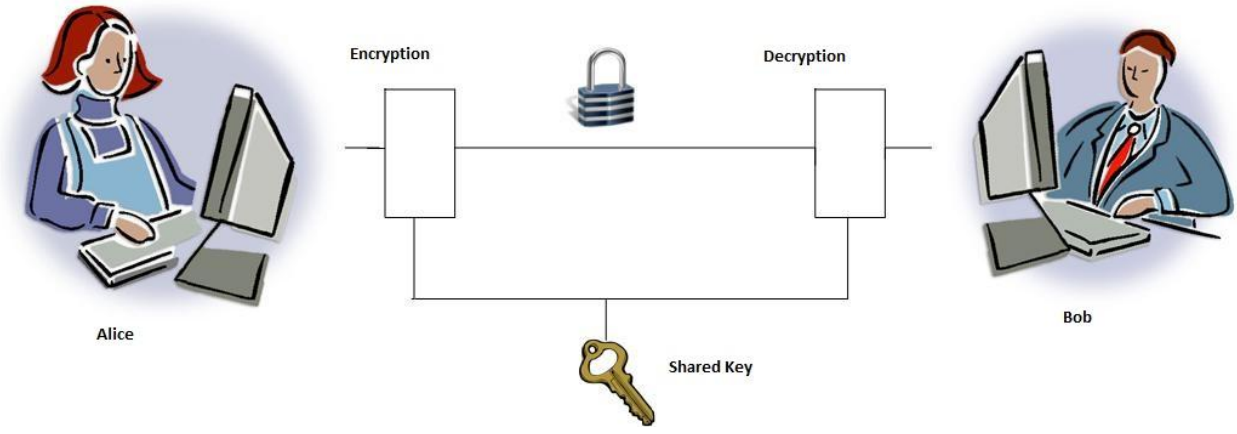


Figure 3: Symmetric encryption where Alice encrypts with secret key K and Bob decrypts with the same shared secret key K

From Figure 3, Alice uses her secret key to encrypt the message and transmits to Bob. Bob, as a receiver decodes or decrypts the message sent by Alice with the same key shared with her. Consider the case, if Alice also wants to communicate with third person Carla, then both of them must agree on their own unique secret key. In this way, a symmetric encryption system needs different and unique keys among all participants. Otherwise, Bob could masquerade as Alice or Carla. Nevertheless, symmetric encryption is very simple to implement.

**Problem with Symmetric Encryption:**

A symmetric encryption needs to establish a unique secret key in advance among all other communicating parties. This approach to satellite communication is annoying since, it should have either pre-shared key or key exchange mechanism. The tradeoff exists in both mentioned mechanisms. Pre-shared key mechanism will be destructive if the key is compromised and the key exchange mechanism adds the complexity in the CubeSat system.

**Asymmetric Encryption**

In an asymmetric encryption, the keys used for encryption and decryption are different. For instance, if Alice wants to setup Bob a secure communication, Alice first asks Bob for his public key, which can be transmitted over an unsecured connection. Alice then encodes a secret message using Bob's public key. This message is secure because only Bob's private key can decode the message. To authenticate herself to Bob, she needs only to re-encode the entire message using her own private key before transmitting the message to Bob. On receiving the message, Bob can establish if it was sent by Alice, because only Alice's private key could have encoded a message that can be decoded with Alice's public key. In this way it can protect authentication of source to some extent.

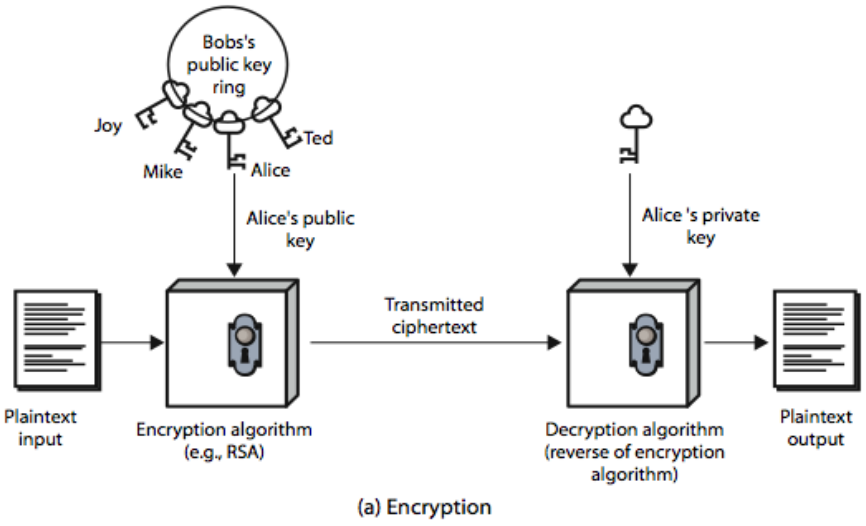


Figure 4: Asymmetric encryption and decryption, taken from [8]

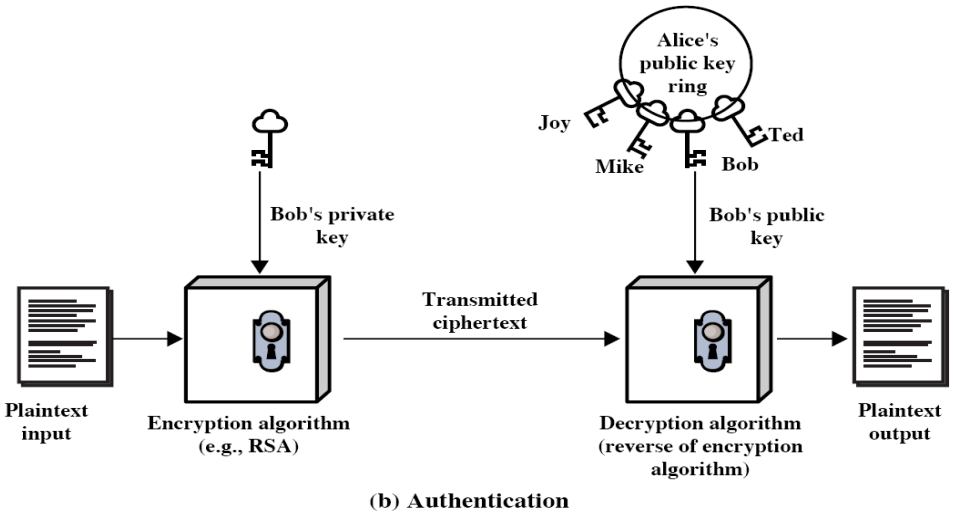


Figure 5: Authentication in asymmetric encryption algorithm, taken from [8]

### Problem with Asymmetric Encryption

Alice must request Bob's public key over an unsecured channel. If this request is intercepted by third person Darth, he can supply Alice with his own (i.e., Darth's) public key. Alice will then encrypt the message with Darth's public key, after which she will re-encrypt the result of the first encryption operation with her own private key. Alice will then transmit the encrypted

message to Bob, which will again be intercepted by Darth. Using Alice's public key in conjunction with his own private key, Darth will be able to decrypt and read the message.

The figure below summarizes the entire process of replay process in four steps:

1. Darth intercept message sent by Alice to Bob which contains the request of Bob's public key for secure communication setup.
2. Darth reply with his own public key.
3. Alice will re-encrypt the message with Darth's public key and her own private key and send to Darth, assuming Bob is the receiver.
4. Darth can decrypt and read the message.

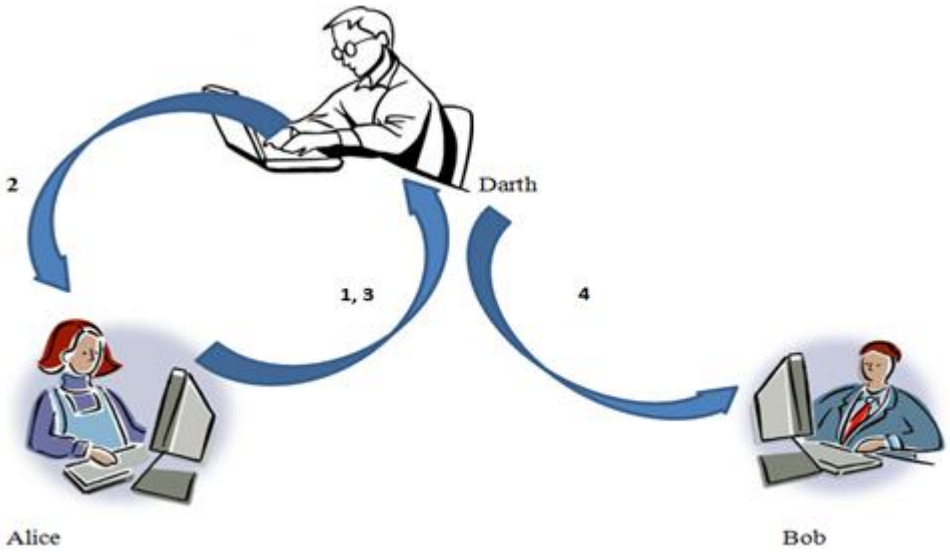


Figure 6: An example of man in the middle attack in which Darth intercepts the communication between Alice and Bob

In above figure, it is possible for Darth to masquerade the communication between Alice and Bob. Consider a scenario where Darth after viewing or modifying the message from Alice, re-encrypt with Bob's public key and transmit to Bob. In this case, Bob has no idea about the originality of message. This type of tampering and resend of message is called replay attack. In CubeSat, this type of attack could be dangerous since anyone who gets accessed can send the forge command to CubeSat. Also, asymmetric encryption is slower than symmetric encryption.

It requires far more processing power to both encrypt and decrypt the content of the message which is not feasible approach in CubeSat [17].

When deciding on which encryption method to use, designer must consider the value of the data being transmitted, the purpose of the transmission, and the technological and computational limitations of the target satellite. CubeSat is not commercial satellite, it is purely lunched for educational purpose, and thus the risk of attack is low. However, anybody can harm the satellite if the security is compromised.

**Need of Integrity and Authentication**

Another major problem associated with satellite encryption is related to the identity and authenticity of the sender. A satellite needs to confirm that the control signals and commands it is receiving from the ground are originated from a legitimate source, this is source integrity. Otherwise, attacker may tamper with bad commands and disrupts the functioning of satellite. Encryption provides only confidentiality not the integrity of the message [29]. A mechanism should be used to protect data integrity. Encryption cannot protect against the replay attack, it only make information opaque such that attacker cannot read the content. Therefore, authentication is needed to avoid spoofing and unauthorized access. Integrity of data should be checked for the telecomands. Thus, in telemetry integrity is highly considered than encryption.

**Hash Function, MAC and HMAC**

Hash function is an algorithm that maps the message of arbitrary length to a smaller set of fixed length. The value returned by a hash function is called hashes or hash value. A Hash value is a checksum that calculated upon a file on piece of data to produce a unique value. Hash value can be used to verify that a file has not been altered after the hash was generated. Collision resistant and one way are the two important properties of function that should be satisfied to be a good hash function [8]. These properties explain that the two different inputs to hash function do not produce a same output. This implies the fact that every input has unique hash output. These properties of hash function allow using the hash values as digital signatures.

**Message Authentication Code (MAC)**

Encryption provides the confidentiality of the message but cannot ensure that a message coming from the alleged source has not been tampered. Authentic encryption can be achieved by combining encryption algorithm with authentication algorithm [29] Message authentication protects the integrity of a message, validates identity of originator, & non-repudiation of origin [8]. MAC is similar to encryption but may not be reversible.

$MAC = C_K(M)$

Where,

M – Input message

C – MAC function

K – Shared secret key

MAC – Message authentication code

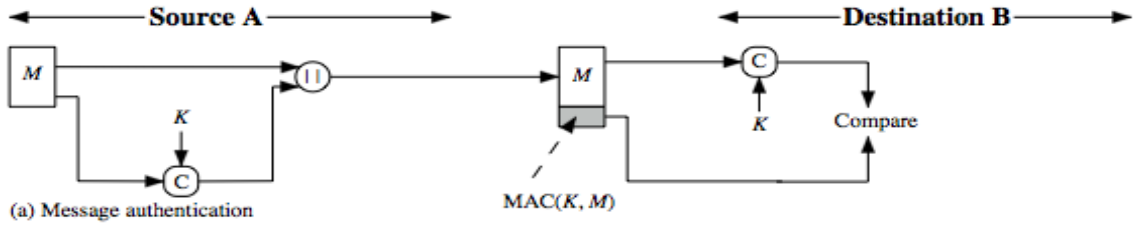


Figure 7: Message authentication process, taken from [8]

**Hash-based Message Authentication Code (HMAC)**

A MAC mechanism which is based on cryptographic hash functions is called HMAC. HMAC can be used in combination with any cryptographic hash functions like MD5, SHA-1, SHA-2, Whirlpool, Tiger, HAVAL etc. It takes underlying hash function as a black box. Figure 8, shows a generation of HMAC using any of these hash function.

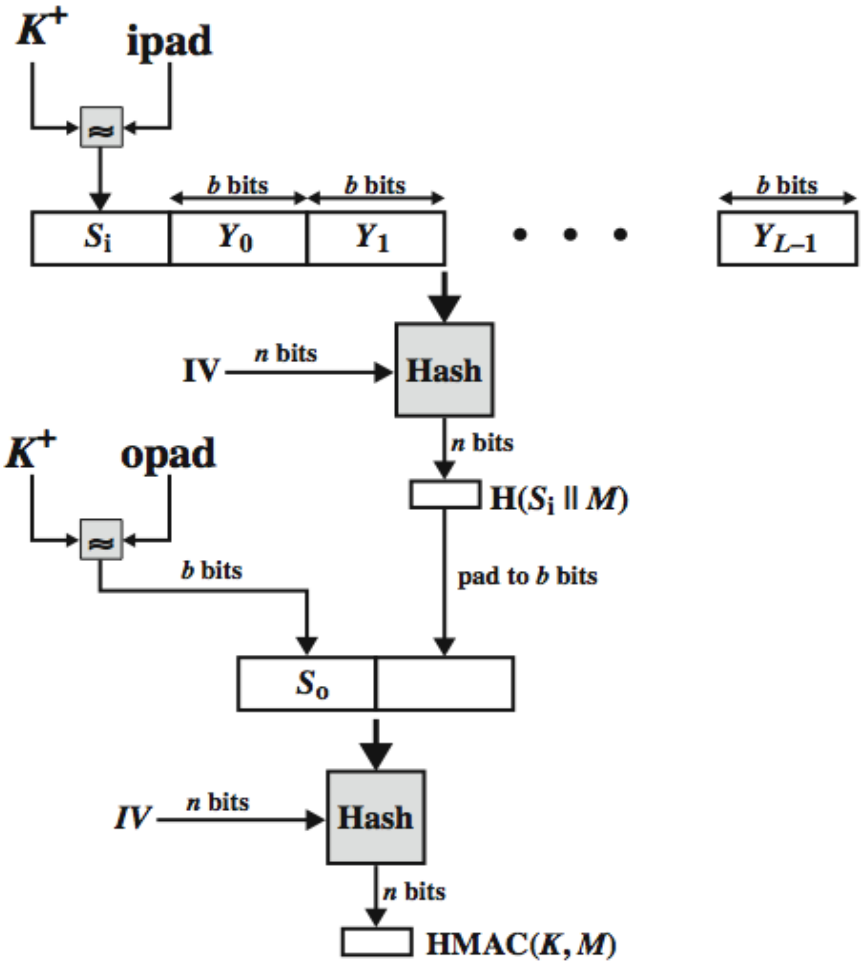


Figure 8: HMAC generation, taken from [8]

$HMAC_K$ (M) = Hash [(K+ XOR opad) || Hash [(K+ XOR ipad) || M)] ]

Where,

K+ is the key padded out to size

opad, ipad are outer and inner specified padding constants respectively.

The key length for HMAC can be of any size; longer keys are first hashed and used whereas shorter keys are strongly discouraged as it would decrease the security strength of the function [9]. HMAC is considered very secure but its security depends on underlying hash algorithm. The output of HMAC has the same size as that of the underlying hash function, although it can

be truncated which is not recommended [9]. Truncation results in fewer bits to transmit over communication link and reduce an authentication algorithm overhead, but it has less security strength than original sequence. Thus, except brute force attack and birthday attack, it is very difficult to destroy HMAC.

CubeSat Space Protocol (CSP) uses Secure Hash Algorithm-1 (SHA-1) as iterative algorithm for HMAC. However, SHA1 is vulnerable to collision resistance (discussed in chapter three) and also slower than other hash functions. In this thesis, we are intending to replace SHA1 with other message digest function which is secure and efficient too.

# CHAPTER 3

# *UPLINK THREAT ANALYSIS*

## 3.1 CubeSat Space Protocol (CSP)

CubeSat Space Protocol (CSP) is a network-layer delivery protocol designed by the students at Aalborg University for CubeSats. The protocol has 32-bit header which contains both network and transport layer information. Its implementation is designed for embedded systems such as the 8-32-bit AVR microprocessor from Atmel but not limited to these microcontrollers. The source code is written in C language and can run on FreeRTOS, POSIX and Linux operating systems. CSP version 1.1 supports Mac OS X and Microsoft Windows too.

Implementation of this protocol is actively maintained by the students at Aalborg University and the spin-off company GomSpace. The source code and other necessary documentations are found under LGPL license and it is hosted by GitHub [12]. CSP corresponds to the same layers as the TCP/IP model. The implementation supports a connection oriented transport protocol (Layer 4), a router-core, and several network-interfaces.
Features of CSP [13]:

- Simple API similar to Berkeley sockets.
- Router core with static routes. Supports transparent forwarding of packets over e.g. spacelink.
- Support for both connectionless operation (similar to UDP), and connection oriented operation (based on RUDP).
- Service handler that implements ICMP-like requests such as ping and buffer status.
- Support for loopback traffic. This can for instance, be used for Inter-process communication between subsystem tasks.
- Optional support for broadcast traffic if supported by the physical interface.

- Optional support for promiscuous mode if supported by the physical interface.
- Optional support for encrypted packets with XTEA in CTR mode.
- Optional support for HMAC authenticated packets with truncated SHA-1 HMAC.

There are two versions of the CSP header. The latest version (1+) was released on November 2010, which supports more hosts and ports. It can be noted that length field is not mentioned in header.

**Version 1+**

In the new version, the header was redefined to support more hosts and ports. CSP now supports up to 32 hosts on the network, with 64 ports available on each host. The port range is still divided into three adjustable segments. Ports 0 to 7 are used for general services such as ping and buffer status, and are implemented by the CSP service handler. The ports from 8 to 47 are used for subsystem specific services. All remaining ports, from 48 to 63, are ephemeral ports used for outgoing connections. The bits from 28 to 31 are used as flag for HMAC, XTEA encryption, RDP header and CRC32 checksum.

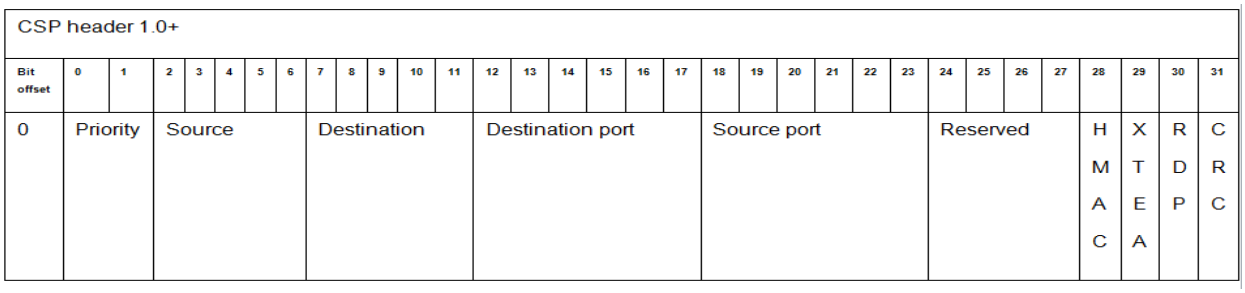


Figure 9: CSP version 1+ header

| CSP Header<br>32 bits | Payload<br>(0-65535) bytes |
|---|---|

Figure 10: CSP Frame

## 3.2 eXtended Tiny Encryption Algorithm (XTEA)

XTEA is the extension of Tiny Encryption Algorithm which was designed by David Wheeler and Roger Needham. Since, some weakness was found in TEA which is referred in [14], XTEA was developed as improved version of TEA. XTEA is 64 bit block cipher with feistel structure and key size of 128 bits. The payload in CSP is encrypted using XTEA during the uplink.

## 3.3 Integrity of message

In the mini-satellite like CubeSat, prime importance is given to the protection of source integrity and data integrity. Source integrity provides the assurance that the data received is as sent by an authorized entity. It gives the confidence to the receiver end that the communication is bind with the legitimate entity. Data integrity gives the assurance that the data received has not been modified in the way. Therefore, integrity protection keeps the originality and authenticity of message. There are several cryptographic techniques used for integrity protection like IPsec, CRC, MAC, HMAC etc. However, HMAC is our interest due to its security strength and feasibility.

## 3.4 Choice of hash function

### 3.4.1 Attacks on SHA1

Secure Hash Algorithm SHA1 is a cryptographic hash function with the output digest size of 160 bits. These generated output is unique thus, HMAC is used in digital signatures, MAC, password table, hash tables etc. In [20], authors proposed a collision in SHA1 in $2^{69}$ hash operations. This attack is about 2,000 times faster than brute force attack which finds the collision in $2^{80}$ rounds. In simple words, probability of occurring same hash output of two different messages is in $2^{69}$ calculations. In contrast, this practically needs quite large time and

energy to obtain this hash value. Therefore, it is not realizable in practical scenario. However, this is not secure enough for future applications since the semiconductor performance doubles in every 18 months (Moore' law) and computer performance per dollar doubles every 21 months (Robert's law) [21]. Furthermore, paper [27] presents an interesting concept of rainbow table which stores all the computed hash. These pre-computed values are used to find the match of hash function and immensely reduce the burden to re-compute hash. Therefore, this method is very efficient for an attacker and saves a huge amount of time.

The hash of data is a prime component in message signing and validating process. In this scenario, if any two different messages hashed to a same digest value, then it is possible to execute unwanted command by a CubeSat. This should be considered or guaranteed in designing a secure CubeSat transmission. Interestingly, we have different alternatives for SHA-1 like SHA-2, whirlpool, RIPEMD etc. However, with the recommendation of more secure and efficient hash function like Blue Midnight Wish (BMW-256) or HAIFA-EDON-R256 (Edon-R256); we are intended to replace SHA1.

### 3.4.2 Alternative digests function:

**BMW and Edon-R**

BMW [22]and Edon-R [23] are the cryptographic hash functions. These hash functions were submitted as a candidate for SHA-3 hash competition organized by National Institute of Standards and Technology (NIST). BMW and Edon-R are a cryptographic hash function with output size of 224, 256, 384 or 512 bits. In this thesis, BMW-256 and Edon-R256 are being discussed for NUTS project. Both the hash functions are resistant against length extension attacks and resistant against multi collision attacks [22]and [23]. Though some attacks have been carried out in BMW and Edon in [24], it is not feasible in current technology. Another reason for selecting these hash functions was computational efficiency. These hash functions are designed to be much more efficient than SHA-1 and SHA-2, with higher level of security. Faster the algorithm, efficient it is from power economic point of view. Power consumption is one constraint that should be considered in CubeSat. In NUTS, photo voltaic cell provides the

energy whereas external batteries are used for backup. It could be interesting to design a system with power efficient in satellite segment.

### 3.4.3 Efficiency and code size comparison

In this section, the results of all three hash functions are presented. All these hash functions were used to generate HMAC with key 20 byte and 64 byte key and input message of different size. They clock frequency used was 16 MHz for AVR UC3 [11]. The clock cycle was initialize to zero before the start of HMAC and marked after HMAC is generated. Here, input message was taken less than 1024 bytes to observe because CubeSat has small data bytes to hash. Also, 20 bytes and 64 bytes of key size were originally provided with the source code and we believe they are strong enough against birthday paradox. Therefore, common inputs for all the hash functions were fed. The average value of output clock cycles were obtained in the table and cycles per byte were calculated. The outputs for 128 and 256 bytes are highlighted since AX.25 frame format can send maximum packet size of 256 bytes. Therefore, input and output of 128 and 256 bytes message for all three hash functions are marked.

This source code was compiled with AVR studio 5.0 in Window 7, 32-bit operating system. The processor of Intel core2 duo with 2.27 and 2.26 GHz with the ram of 4 GB was used.

Table 1: Computational speed of Edon-R256 for different input and key size

| Edon-R256 | | | | |
|---|---|---|---|---|
| Size of input message | Average cycles (for 20 byte key) | Cycles/byte | Average cycles (for 64 byte key) | Cycles/byte |
| 8 | 8920 | 1115 | 9362 | 1170.25 |
| 16 | 8876 | 554.75 | 9318 | 583.375 |
| 32 | 8803 | 275.09 | 9246 | 288.93 |
| 64 | 10515 | 164.29 | 10958 | 171.21 |
| 128 | 12071 | 94.30 | 12512 | 97.75 |
| 256 | 15179 | 59.29 | 15620 | 61.02 |
| 512 | 21395 | 41.78 | 21836 | 42.64 |
| 1024 | 33827 | 33.03 | 34268 | 33.46 |

Table 2: Computational speed of BMW-256 for different input and key size

| BMW-256 | | | | |
|---|---|---|---|---|
| Size of input message | Average cycles (for 20 byte key) | Cycles/byte | Average cycles (for 64 byte key) | Cycles/byte |
| 8 | 14158 | 1769.75 | 14600 | 1825 |
| 16 | 14114 | 822.13 | 14556 | 909.75 |
| 32 | 14041 | 438.78 | 14484 | 452.65 |
| 64 | 16155 | 252.42 | 16598 | 259.34 |
| 128 | 18081 | 141.25 | 18522 | 144.70 |
| 256 | 21929 | 85.66 | 22370 | 87.38 |
| 512 | 29625 | 57.86 | 30066 | 58.72 |
| 1024 | 45017 | 43.96 | 45458 | 44.37 |

Table 3: Computational speed of SHA-1 for different input and key size

| SHA-1 | | | | |
|---|---|---|---|---|
| Size of input message | Average cycles (for 20 byte key) | Cycles/byte | Average cycles ( for 64 byte key) | Cycles/byte |
| 8 | 14563 | 1820.37 | 14951 | 1868.87 |
| 16 | 14256 | 891 | 14760 | 922.5 |
| 32 | 14322 | 447.56 | 14538 | 454.31 |
| 64 | 16387 | 256.04 | 17605 | 275.07 |
| 128 | 18670 | 145.85 | 19833 | 154.94 |
| 256 | 22062 | 86.178 | 23770 | 92.85 |
| 512 | 30921 | 60.39 | 31206 | 60.94 |
| 1024 | 45634 | 44.55 | 45896 | 44.82 |

Form the information obtained from three tables, two graphs are drawn below which demonstrates the computational efficiency of all three hash functions to generate HMAC using 20 byte and 64 byte key. The horizontal axis represents the size of message hashed and the vertical axis shows the time required to hash relative to cycles per byte.

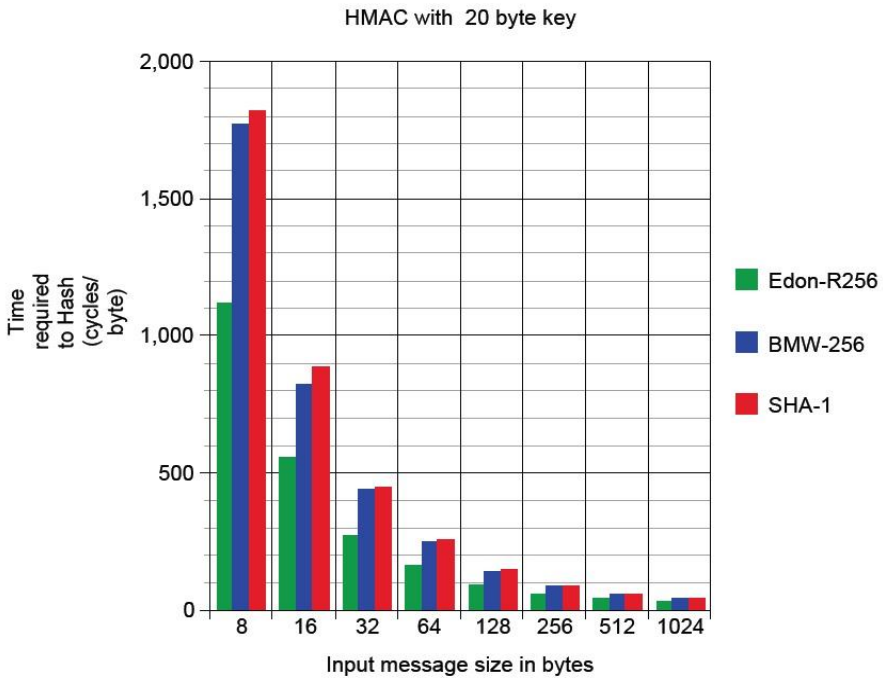


Figure 11: Performance of Three hash functions in HMAC using 20 byte key

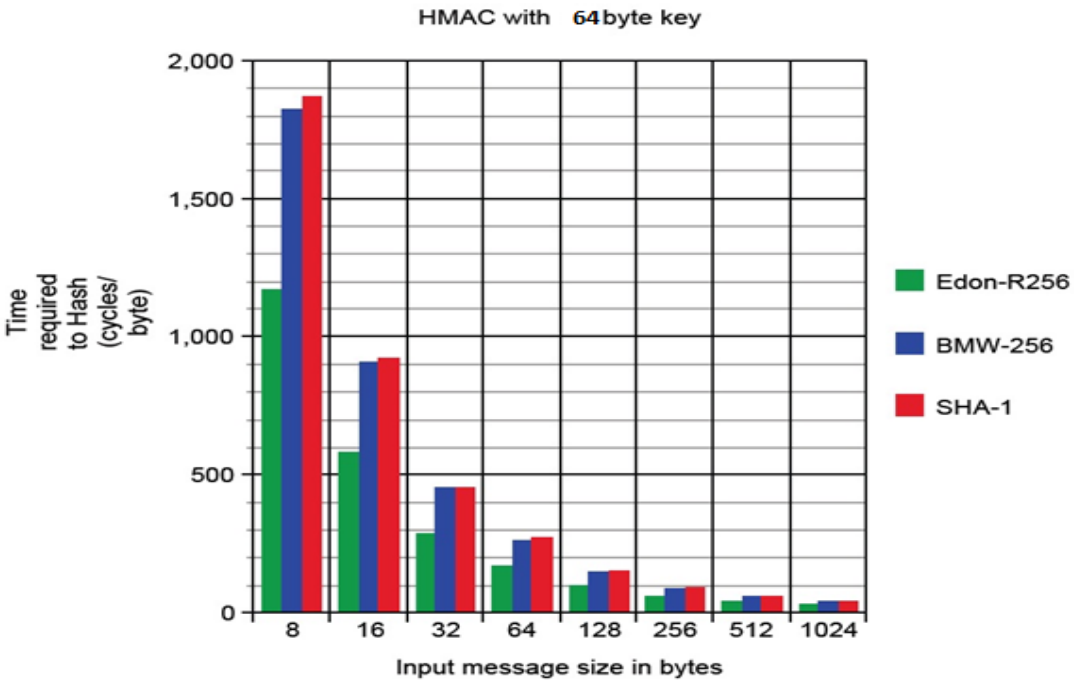


Figure 12: Performance of Three hash functions in HMAC using 64 byte key

In figure 11 and 12, bar with green color represent the computational speed of Edon-R, blue represents BMW and red represents SHA-1 against the different message size. From figure, Edon-R 256 and BMW-256 are faster than SHA-1 for all the input bytes. It also shows that Edon-R256 is much faster than BMW256 and SHA-1 in all formats. An important conclusion can be drawn from the graph is that with the increase of input message size, the efficiency of hash function increases. Another remarkable conclusion is the size of key, larger is the key size slower is the computation of HMAC. Similar tests are being carried by VAMPIRE in different platforms and results obtained are analogous to our result. VAMPIRE develop a toolkit called SUPERCOP (System for Unified Performance Evaluation Related to Cryptographic Operations and Primitives) to measure the performance of cryptographic software [38]. This toolkit benchmark the time to hash the different sized data. In [34] and [35], the different hash functions are compared and presented in tabular form and graph. Their output for different processors and frequency are congruent to the result incurred from our analysis.

After compiling the source code, hex file is obtained for respective hash functions. These hex files can be used to determine the size of source code. For these functions, Edon-R256 has the smallest size of 40 KB, followed by SHA-1 which has 74 KB and finally BMW which has 92 KB. Since, these hash functions were not designed for AVR platform; they still can be optimized for AVR platform in term of execution efficiency and code size.

### 3.5 Cyclic redundancy check for error detection

A 32-bit checksum is used in CSP to detect an error in transmission. A sequence is obtained as the remainder of polynomial division of their contents; these values are similar at both the transmitting and receiving end for error free transmission. It is simple to implement and good algorithm in detecting error caused by noise. It keeps the message of original transmitted message therefore keeps an integrity of message. However, CRC is not suitable for protection against the intentional professional hacker. Most of the launched CubeSat used CRC due to its simplicity and less overhead in transmission.

## 3.6 Secure Key Exchange Mechanisms

We have already discussed the cause of replay attack in earlier chapter. Replay and tampering is possible if an attacker has an access to a key. Strong encryption algorithm or strong hash algorithm does not remain secure if their key is compromised. Thus, secure key exchange mechanism is foremost for secure communication. In NUTS, same key has to be shared between ground segment and satellite segment. This section analyzed the possible efficient mechanism to share the key between two parties. Here are some commonly adopted approaches for secure key exchange between two parties.

### 3.6.1 Symmetric Key Distribution

Some common forms of symmetric key distribution are describes in this section. This approach uses same key between two communicating parties and most of them uses Trusted Third Party to assist key.

**Wide-Mouth Frog**

This is symmetric key management protocol and involves Trusted Third Party (TTP) to distribute a new shared key. It also protect against the replay of old messages by providing the timestamp. Two parties wishing to establish a secret session has to share a unique key with TTP. However, this protocol suffers from following problems:

- A is not assured that B exists
- Insecurity while distributing initial key between TTP and other party
- Initiator must be able enough to generate strong keys
- Synchronization of clock is needed to generate timestamp

**Kerberos**

It provides mutual authentication between clients and servers in distributed environment and also protects against replay and eavesdropping [8]. It uses two TTPs, a ticket granting server and authentication server. After authentication of client, ticket granting server issues a ticket

and session key to client. Interaction between clients and server is protected with key and ticket which has fixed life time. This protocol provides mutual authentication between client and server, timestamps to protect against replay. There are still some problems associated which are as follows;

- Since it uses timestamps, thus synchronization is needed.
- It needs the continuous availability of central server.
- Authentication server and ticket granting server can be a target for DOS attack.

**Otway Rees**

This is also authentication protocol which uses TTP for shared session key. Mutual authentication is done between party A and B and also uses index number to protect against replay. All these interacting messages are encrypted thus gives more security. Some problems related to this method are follows;

- Secure generation and distribution of key between communicating parties and TTP is needed.
- During authentication process may leads malicious intruder to access shared key.

**Needham-Schroeder**

This protocol is primarily an authentication protocol in first phase. After authentication is completed, a shared symmetric key is established between the parties. This protect against replay by the use of random numbers that provides the freshness of message. This protocol uses symmetric cryptography which makes it faster. Some limitations of Needham-Schroeder are listed below [8];

- Replay attack is possible if old session key is used.
- TTP is a single point of failure.
- TTP must be trusted by both the parties.
-

**Blom's Scheme**

This is based on symmetric key pre-distribution scheme and does not need third party. The security of Blom's scheme depends on the chosen value of "t" which is called Blom's secure

parameter. Larger the value of "t" higher the resilience but increase the amount of memory required to store the key.

- Tradeoff lies between key size and memory requirement.
- In order to prevent share key computation by attacker, all the sets of parameters selected by user should be linearly independent.

**Decentralized scheme**

In this scheme, only two parties are involved in secure key exchange. These are based in challenge response based. Therefore, one party acts as initiator and another act as responder.

This scheme is simple to implement and does not need third party for key distribution. However, it is not free from limitations which are listed below.

- Risk of attack is high.
- Attacker can intercept and masquerade.

### 3.6.2 Merits and demerits of Symmetric Key Distribution

Several symmetric key exchange and key distribution schemes have been explained in previous section. The advantages and disadvantages have been found for all schemes, some common of them are listed below.

**Advantages of Symmetric Key Distribution**

- It is easy to manage entities in network and easy to change entity's keys except for Kerberos.
- Only one permanent key is stored.
- Symmetric-key ciphers are faster.

**Demerits of Symmetric Key Distribution**

Most of the key exchanges mentioned make use of a Trusted Third Party, where every entity shares a pre-established initial key with the TTP. Therefore, in every communications require initial contact with TTP. Here are the lists of issues associated with TTP [39]:

- "Trusted Distributor" problem: the secure communicating channel and the trusted distributor are needed and TTP is the single point of failure.
- TTP must store keys for all the communicating parties of parties.
- Secret keys may be compromise during transmission
- TTP can read all messages therefore trust is required by all entities.
- Since TTP is center for all communication parties, it shows performance bottleneck.
- Lots of communication through TTP makes it target for attack.

### 3.6.3 Asymmetric Key Distribution

Asymmetric or public key cryptosystem require two separate keys known as public key and private key for each user. Public key is known to all where are private key is secret to user. Public key is used to encrypt the message and corresponding private key will be able to decrypt that message. Unlike, symmetric key algorithm, an asymmetric algorithm does not need to establish initial session to exchange secret keys between communicating parties.
Some schemes of public key distribution are discussed below.

**Diffie-Hellman Key Exchange**

This method allows two unknown parties to jointly establish a shared secret key over an insecure public network. Here session key is only created when it is needed, so it is not necessary to store it. This decreases the risk of exposure of key. There are some problems listed in this approach which is listed below.

- Likelihood of man in the middle attack, since communicating parties is not authenticated.
- Its strength depends on the choice of prime numbers.

**ElGamal Key Agreement**

This algorithm is based on Diffie-Hellman key exchange. It consists of encryption algorithm, decryption algorithm and key generator component. The security of ElGamal depends on underlying cyclic group and padding scheme used on the message. The message can be

encrypted to many possible cipher texts which add confusion to the attacker. Some problems associated with this algorithm are mentioned below.

- ElGamal key exchange is slow.
- Increase the overhead since the size of cipher text is twice than that of original message.
- No authentication of entity.

**Shamir's Three-Pass Protocol**

This protocol allows one party to securely send a message to a second party without the need to exchange or distribute encryption keys. The sender and the receiver exchange three encrypted messages in three phases. Problem associated with this public key distribution protocol is man in the middle attack since there is no authentication of entities.

**Digital Signature Algorithm**

The DSA signature scheme has advantages, being both smaller and faster, over RSA [8]. DSA cannot be used for encryption or key like RSA. Nevertheless, it is a public-key technique. The DSA is based on the difficulty of computing discrete logarithms, and is based on schemes originally presented by ElGamal and Schnorr. But, this algorithm is slower than symmetric approach.

### 3.6.4 Merits and demerits of Asymmetric Key Distribution

Some of the common advantages and disadvantages of public key distribution are listed.

**Merits of Public Key Crypto for Key Exchange**

- Asymmetric key cryptography offers additional features which are not easily obtainable with symmetric cryptography

  -Online TTP is not required

  -Source integrity and authentication
- The number of keys necessary in large network may be considerably smaller than in the symmetric-key environment.
- Only the private key should be kept secret.

**Demerits of Public Key Crypto for Key Exchange**

- Public key distribution methods are slow compared to symmetric cryptography.
- Key sizes are typically much larger than those required for symmetric-key encryption.
- An encrypted message can only be sent to a single recipient.

### 3.6.5 Why Symmetric key and Why not Public key?

Public key encryption/decryption algorithm is a fundamental and widely used technology around the globe. But it is complex, slower and consumes more power. Therefore, it is not common among ultra-low power environments like wireless sensors and CubeSat [19]. Therefore, Symmetric key encryption algorithm with low-energy consumption is used in low power environments. Furthermore, public key encryption is slower than symmetric encryption of same security level. It needs more computation time for the computation of public and private keys. From this scenario, symmetric key is considered as simple, fast and low power consuming methodology. Moreover, the important scenario of CubeSat should be accounted while designing or selecting the key exchange protocol. The propagation delay between earth segment and space segment makes the protocol with several authentication steps and multiple round trips are not feasible to use in CubeSat. Since, probability of interference due to noise is higher in long distance wireless transmission, Signal to Noise Interference (SNI) decreases resulting in error in transmission. Multiple round trip authentication mechanism has to setup handshake before establishing communication between two parties. Therefore, multiple round trips have high probability of getting error which results in authentication failure. Similar problem may appear for synchronization which results in denial of service.

## 3.7 Countermeasures against replay

It is clear that we are not encrypting the uplink payload. Our intension is to protect the integrity and replay messages so that tampered and replayed payload is rejected at the satellite segment. We mentioned HMAC to protect integrity of message but it fails to protect against replay. For instance, an attacker with an idea of reboot command of CubeSat can retransmit again and

again which makes the satellite continuously restarting resulting into DoS. This section discusses some techniques used in communication to prevent form replay. One time passwords, session tokens are mostly used in web based client server authentication model. One Time Password (OTP) is used to authenticate only once and renewed for next authentication. Similarly, session tokens are the data that carries the information for that communication session. These two concepts protect replay attack since they relies in non-reuse of key. So, in future an attacker cannot use the previous password or session token to establish a communication. Besides these techniques, uses of time stamp and sequence number are other techniques widely used in communication systems against replay attack.

## 3.8 GENSO

Global Education Network for Satellite Operations (GENSO) is a software standard developed with the goal to release a global network of university ground stations and radio amateur to support the operation of CubeSat. The interconnection between ground stations is done via the internet and software allows to share and free access to other station. Therefore, it allows commanding their CubeSat from other part of world [40].

The visibility of CubeSat is about 20 minutes in a day, in rest of the time ground station is in ideal state. In this scenario, an idea of GENSO is very innovative towards the utilization of station from any part of the world and also, to get access similar data from other CubeSat. GENSO has three components Ground Station Server (GSS), Authentication Server (AUS) and Mission Control Client (MCC). Ground station operator run GSS and MCC is run by mission controller. GSS after receiving the data notifies AUS. AUS is center core for authentication and control access control among the permitted bounded entities. Now, AUS after receiving notification from GSS pass the notification to MCC owning the satellite. MCC then retrieves all the locally stored data from GSS.

NUTS is also planning to use GENSO but till this date the software has not been released. It is very important to study its access control and authentication measure before adopting the new software. Nevertheless, this software seems very efficient in term of utilization of CubeSat.

# CHAPTER 4

## *PROPOSED CSP FOR SECURE UPLINK IN NUTS*

This chapter finds the best approach for the NUTS access control and uplink security within underlying constraints. The modified CSP is presented along with its merits and demerits. We are not using any encryption mechanism on uplink payload which will only add complexity in CubeSat system.

### 4.1 Key size and hash function

We propose symmetric key distribution, 20 byte key hardcoded in both ground segment and satellite segment. The key length of 20 bytes has good resistance to birthday attack and strong enough for brute force attack. The calculation shows that 20 bytes (160 bits) hash has approximately $1.46*10^{48}$ different outputs. If these are all equally probable then it will take around $1.5*10^{24}$ attempts to generate collision using brute force. It was also observed that 20 bytes key was faster than key length of 64 bytes to generate HMAC.

Single key is used for HMAC, though we discussed about the different key exchange mechanism, it is not suitable to use either TTP or any protocol using multiple RTT. Propagation delay and loss of signal makes the authentication process complex and also generates the resynchronization problem. It is difficult to resynchronize the key in robust wireless environment and may leads to denial of service. So, HMAC code protects the system against integrity but adds an overhead in communication system and complexity of calculating as well as comparing the code in satellite segment.

## 4.2 Countermeasures against replay

Sequence number in other hand solves the problem of replay attack. Sequence number is the random sequence that adds the freshness in every message and keeps the information about following message. Attacker cannot replay with old sequence number which is already registered in the receiver's entry. Also, if an attacker tries to insert the new sequence then he could not modify the HMAC content. HMAC is very strong cryptographic function which produces fixed byte output depending upon the key used. The data is initialized; sequence number is added and encapsulated with CSP header to form CSP packet. HMAC is calculated upon this complete CSP packet and concatenated with CSP packet. CRC is calculated upon entire packet (from CSP header to HMAC code) and appended with this packet to form payload. This final block is ready to send using AX.25 protocol. This complete procedure is shown in the figure 13.

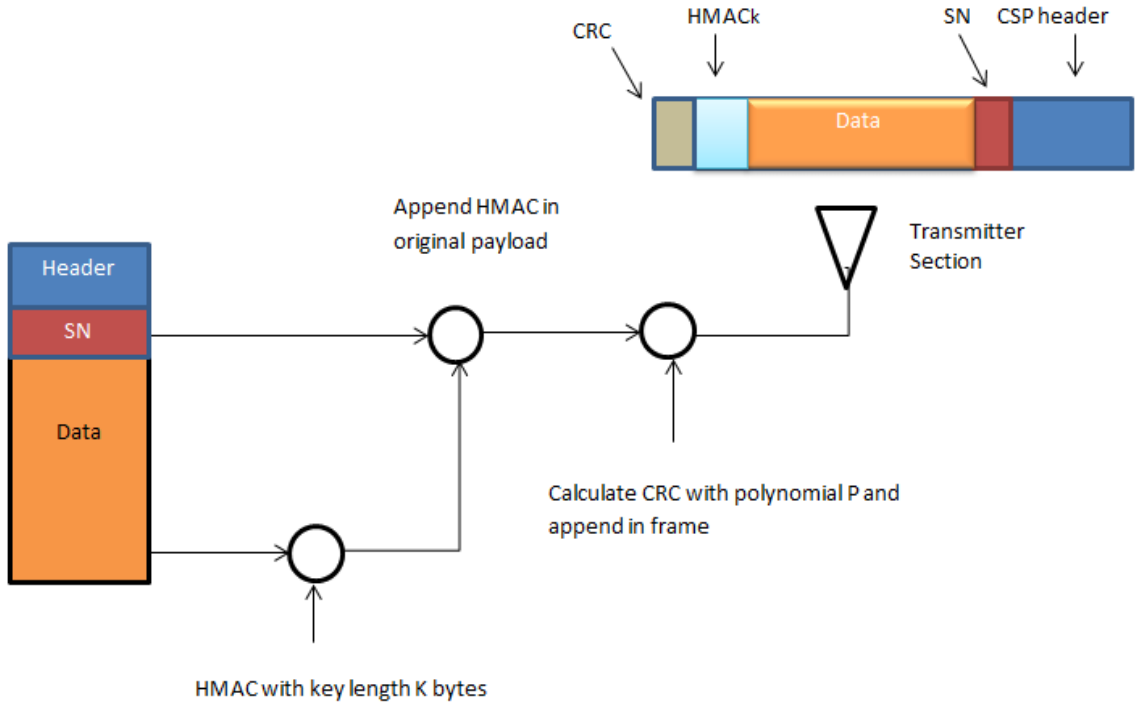


Figure 13: Proposed NUTS Secure Uplink

This complete frame at transmission section shown in Figure 13 is encapsulated in AX.25 radio link protocol and transmitted towards receiver's end in form of radio packets. In the receiving end, the packets are received and processed. Figure 14 shows the complete steps of information extraction process in proposed secure NUTS receiver.

In order to eliminate confusion regarding the term used for packets, we assume them as follows;
CSP information field: Data
Complete CSP packet: CSP header+ SN+ Data
CSP payload: CSP header+ SN+ Data+ HMAC code+ CRC code
Complete AX.25 packet: AX.25 frame/ radio packet
AX.25 information field: Information field (CSP payload)

Figure 14 shows the proposed secure transmitter for NUTS. It is assumed that the radio packets are received and buffered in initial condition, and then the operation of system is explained as follows; get the radio packets from the buffer which is AX.25 radio frames. This radio packet constitutes of head, information field and tail bits. Information field encapsulates CSP payload which is extracted in the next step. AX.25 tail bits contain 16 bit frame check sequence to determine the error on frame during transmission. Upon detection of error, the frame is discarded and computes for the new frame. The next security check computes for CRC-32, compare with the received CRC code and makes a decision. Similarly, HMAC of complete CSP packet is calculated and compared with the received HMAC code. An unaltered sequence will have same code and is proceed further but the packets are discarded if different values are detected. Upon successful verification of HMAC, sequence number is checked against replay. The packet is discarded again if sequence number is already registered in receiver's entry. If the fresh sequence number is found then packet is further processed and sequence number is updated. The commands are extracted and sent to other subsystem and the system again starts extraction with new radio packet. Also, after rejecting packets, system will start with extraction of fresh packet.

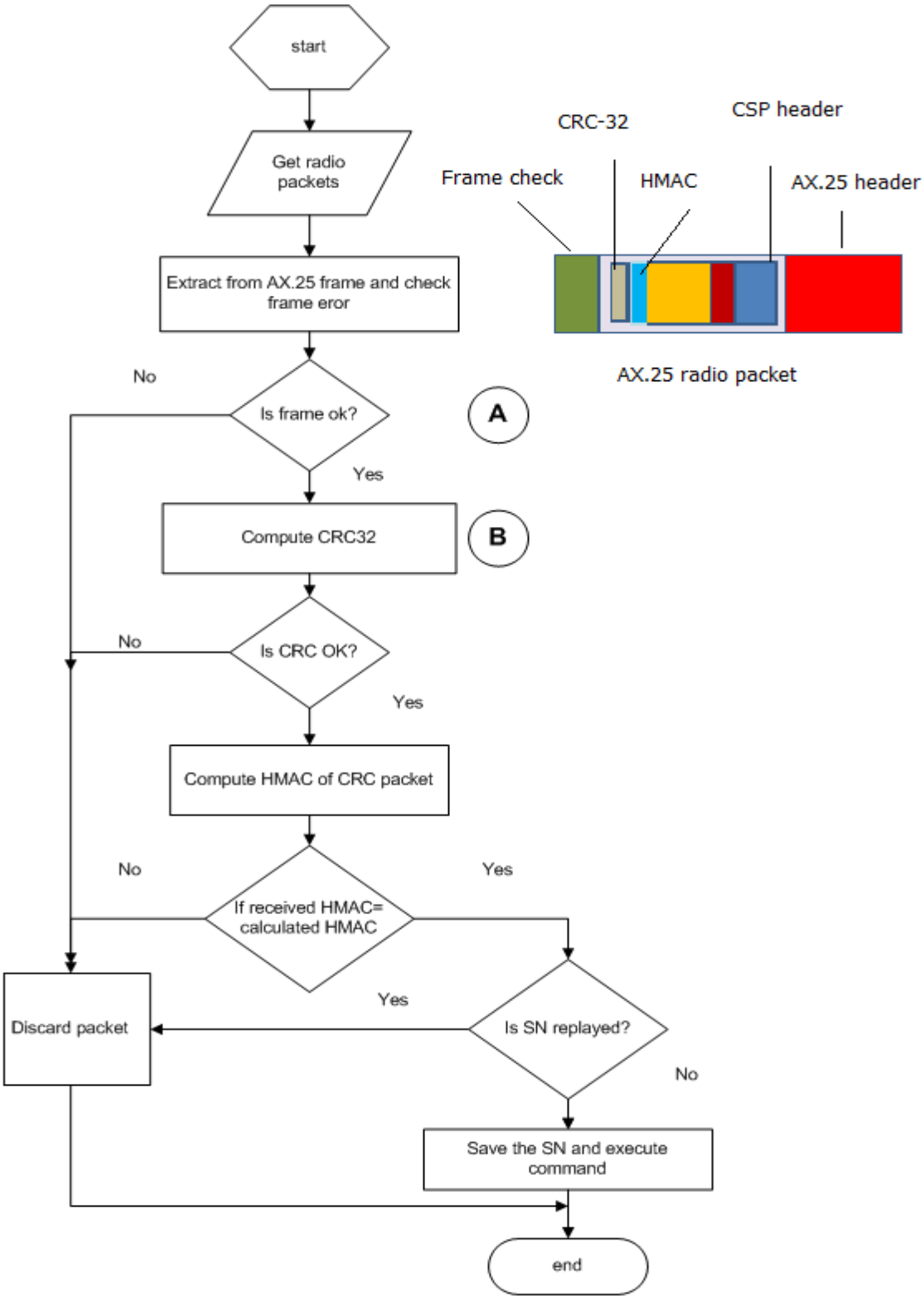


Figure 14: Proposed NUTS secure receiver

The replayed sequence is identified in almost last stage which is the primary disadvantage of this design. Therefore, if an attacker releases numerous replayed packets then all these packets are able to execute up to almost final stage. This will only engage the processor and results worthless with loss of energy and resources. But this issue can be resolved prioritizing the SN checking mechanism which can be placed between A and B in Figure 14 . By prioritizing the SN check results another priority issue for HMAC. Here again same problem appear for HMAC which is checked at the last stage therefore altered packed are executed up to last stage. The use of CRC prevents packets from being alter but cannot prevent professional hacker to alter the original message. Therefore, this tradeoff should be considered during the design.

## 4.3 Modification in CSP header

CSP header 1.0+ (Modified)

| Bit offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Priority | | Source | | | | | Destination | | | | | Destination port | | | | | | Source port | | | | | | Reserved | | | SN | HMAC | XTEA | RDP | CRC |

Figure 15: Modified CSP header, SN flag used in 27th bit

Figure 15, shows the modified proposed CSP header on version1.0+. In this modified header, SN flag is incorporated as 27th bit; this bit was reserved for future purpose. The enable of this flag indicated the presence of sequence number in the packet. In NUTS, every CSP packet will be consisting of sequence number. New fresh sequence number will be assigned in each new CSP packet.

## 4.4 Length of Sequence field

We have proposed to use the sequence number of 16 bit long. It gives 65536 unique counts which is enough to send the unique number with each CSP packet till life time of NUTS. It is assumed that the life time of NUTS ranges from 2 month to maximum of 2 years. From the simulation by NUTS member, earth segment can communicate 28 times in a week with satellite station. Also, multiple commands are uploaded at a time, in this scenario; 16-bit sequence is quite realistic to provide unique identity for each CSP frame.

## 4.5 Sequence number synchronization

It is necessary to synchronize a sequence number after the reboot of system in satellite. Also, consider the worst case when all the transmitted commands from ground station are lost on the way to satellite. Then, transmitting a set of new commands next time will have new sequence number while satellite segment has previous old SN. This type of unusual leaped sequence may be rejected by the system, resulting in denial of service. Therefore, a command is necessary to synchronize the SN and keep it updated. Therefore, to avoid the rejection of all packets due to mismatch problem in SN a special command request "SYN_NEW_SN" is sent that does not has sequence number. This command makes the OBC of satellite components to fetch the latest sequence number used and send as response towards ground station. Similarly, this concept is also useful in reboot command. We can set a reboot command without SN such that synchronization would not be necessary, this makes the reboot efficient since no computation for SN is required and no synchronization is checked.

**Request format**

The request and response command forma are drafted by NUTS members. To synchronize the sequence number we have proposed two commands which are discussed below.

Table 4: Request command format from NUTS draft

| 16 | 48 |
|---|---|
| **Command type** | **Command argument** |

Table 5: Proposed Request command description

| **Request type** | **Description** | **Parameter** |
|---|---|---|
| SYN_NEW_SN | Synchronize with received sequence number | 16 bit fresh sequence |

**Response format**

On successful reception and handling of a request, the success bit will be set high in the response data and send the latest updated sequence number to the ground station. With the idea of latest SN, ground station can then transmit CSP packet with new sequence.

Table 6: Response format from NUTS draft

| 1 | 7 | 1 | 7 | 24 | 2^24 |
|---|---|---|---|---|---|
| **Success bit** | **Command type** | **Last response** | **Options** | **Length** | **Payload** |

Table 7: Proposed response description

| **Response type** | **Description** | **Parameter** | **Parameters** |
|---|---|---|---|

| | | | |
|---|---|---|---|
| RES_GEN_ERROR | General error | N/A | |
| RES_SYN_NEW_SN | Response to synchronize SN command | Fetch latest updated SN | |

Table 7 shows two commands as a response to request command. RES_GEN_ERROR is a response to general error which is already proposed and drafted in NUTS whereas we propose another error response RES_SYN_NEW_SN to distinct from general error. This error indicates that the satellite cannot fetch the latest SN from the entry. In this situation we have no alternative to reboot the system and request for latest SN. All the drafted commands for NUTS are given in an appendix A.

These special commands needs the SN check mechanism at the receiver to distinguish from the normal commands. This mechanism is shown in flowchart of Figure 16. It assumes that the radio packets are received and extracted. Therefore, it only focuses in the validation of SN flag that determines the types of commands.

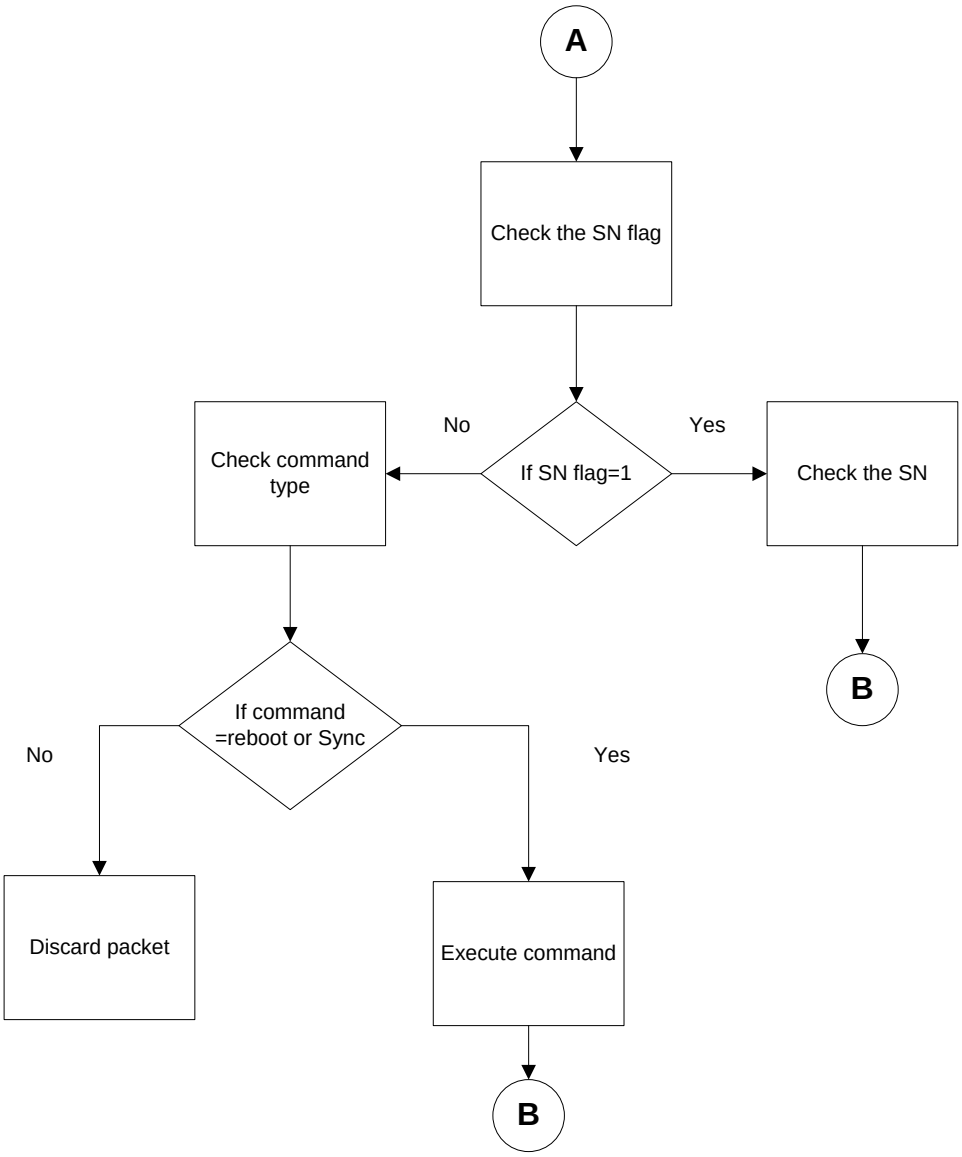


Figure 16: Algorithm checks SN flag to distinguish the types of received command

From figure 16, system checks for SN flag first and if the SN flag is found high then it will check the SN to verify replay. The further processing can be found in Figure 14. However, if the SN is low then it will check for type of command. If the command received is to reboot or resynchronize then it will execute the command as shown in Figure 14 else discard that command assuming bogus data. This is how it prevents bogus packets without SN flag to compute further.

**Effects with the removal of Sequence number**

After the removal of SN in reboot and synchronization request, reboot can be performed without an idea of previous SN and same as for synchronization. The primary advantage of this implementation is in the execution of these two commands.

- With no SN, computation is faster.
- No need to synchronize the SN.
- Probability of rejection of packet due to SN mismatch is decreased.

However, some serious problems arise with the removal of SN which are listed as follows;

- With no SN, an attacker can replay with packets and make system continuously rebooting.
- If attacker has an accessed to a key, then he can request the SYN command to receive the latest SN from the satellite segment and try new SN to execute command. This may leads an attacker to have permanent access on satellite.

The solution to this problem could be the use of time stamp in every packet. In NUTS project, we have not decided to implement time although discussion about the use of absolute and relative time was held. Therefore, after the decision is made to use time in satellite, it could be studied to use the time stamp instead of sequence number against replay.

## 4.6 Frame size considering sequence number

It is clear that our data is encapsulated in CSP frame which is again encapsulated within AX.25 protocol. In this section we have calculated the maximum size of our final radio packets ready to transmit in ideal case.

Modified CSP header: 32-bit
Sequence Number: 16-bit
Command: 64-bit (maximum size)

CRC: 32-bit

HMAC code: 256-bit

Total payload: 32-bit+ 16-bit+64*16(sixteen commands at a time)-bit + 32-bit+256-bit=1360 bits

AX.25 header + flag: 128-bit+8-bit=136 bits

AX.25 information field: 816 bits (maximum information that AX.25 can carry is 2048 bits)

AX.25 tail (frame check sequence+ flag): 16-bit + 8-bit=24-bit

Maximum size of NUTS uplink radio frame=136-bit+ 1360-bit+24-bit= 1520 bits

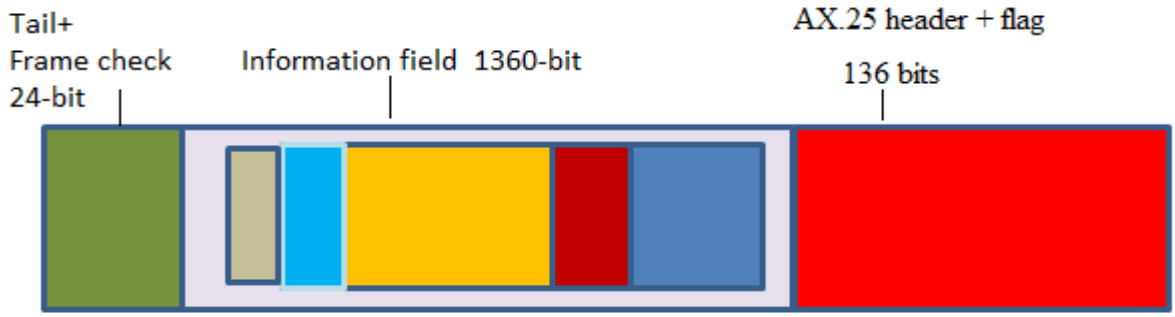


Figure 17: NUTS AX.25 radio packet

## 4.7 Transmitting with Time Stamp

It is clear that time stamp gives strong solution against relay. Also, time stamp does not need synchronization and of sequences between earth and satellite segment. Therefore, the two commands proposed earlier for synchronization request-response is not necessary. The only thing need to consider is the time drift between received and transmitted time. The time margin should be allocated such that the multipath propagated signal can reach the destination within bounded time. The packets received after the time limited could be considered as replay. Therefore, attacker has no chance to replay the packets. If an attacker attempts to modify the time stamp then the HMAC code will be altered which can be detected at the receiving end.

# CHAPTER 5

# *CONCLUSION*

## 5.1 Discussion and conclusion

After the analysis of entire system, this part of report is focused in the important outcome of each task carried through the research. Firstly, we found the integrity protection of message is more significant in CubeSat than its confidentiality. CubeSat is launched with motivation to explore the space and fetch the informative educational information. Therefore, visibility of uplink command to an outsider does not matter unless an outsider tries to tamper it. Thus, to protect the commands from being altered by an attacker was the prime target so we preferred to use HMAC code. Nevertheless, integrity protection and encryption gives very strong protection but adds the overhead in the transmission.

Secondly, we decided to use HMAC code to protect integrity of uplinked message. An attacker trying to tamper the message will result differently HMAC code which can be detected at the receiver end. Here, we analyzed three hash functions to test its efficiency in AVR 32-bit microcontroller platform which was not performed till now. This analysis was completely a new task and the results obtained were very satisfactory. We found that Edon-R was very fast with small code size (hex file) than BMW and SHA1. But security strength of BMW is higher than Edon and SHA1 according to the various documentation of cryptanalysis. It is up to the NUTS authority to decide the choice of hash function for HMAC. In [36]and[37], writer presents a fact that even weak hash function can be used in HMAC and can produce a strong output due to the secure nature of HMAC. We also found the 20 byte key was efficient to produce HMAC code and can provide enough security for NUTS. Regarding the key for HMAC, it can be hardcoded in the memory of microcontroller at satellite segment and earth

segment. With the use of single permanent key, the problem of key distribution and synchronization is avoided but add the risk of key compromise.

Finally, we agreed to use sequence number to provide the freshness in message, it protect against replay. NUTS has a life time of 2 years (maximum consideration), so 16-bit SN is far enough to send each CSP packet with unique number. These sequence number may loss synchronization between transmitter and receiver which may leads to DoS. To prevent this, we added a command that synchronizes the SN. This command will make the OBC of satellite segment to fetch the latest updated (entered) SN from the memory and transmit to ground station. With the idea of this received SN, it is possible to use fresh SN and transmit new radio packet. In contrast, these two commands make system vulnerable to a replay attack which may results continuously rebooting of a CubeSat. We also presented in brief that time stamp could be a good solution against replay attack and to eliminate a resynchronization problem due to SN mismatch. In addition, the use of CRC has significant role in error detection and rejection of fake or corrupted packet such that the further processing is ended. This also saves the time of processor to engage in processing of corrupted packets.

Last but not the least, working with NUTS provides me the opportunity to explore new ideas and analyze the real term project closely. It also helps to understand the real environment, co-operate with the project members from different fields of expertise. Since all of us involved in the NUTS project were from different departments. In addition, we believe that this result will help NUTS authority to adopt the proper security parameters.

## 5.2 Further enhancement

This thesis was a research to analyze the threats and the possible solution to mitigate the threats in NUTS. We hope this report fulfills the target assigned for the secure uplink of NUTS. However, some enhancement can be done on this work. In future, the security of ground station can be analyzed.

Integrity and encryption gives strong security but it adds overhead too, in future this can be a matter of study. If overhead has less effect in transmission then it is better to make system secure. But, significance of these should be determined first in contrast, it is clear that launching CubeSat is getting popular this days and in future more and new version of nano-satellite will be developed and release. May be this will focus and motivates an attacker.

We worked on uploading 16 commands at a time which optimizes the radio resources, still we can upload more commands to optimize radio channel. Till this date, all the command set has not been finalized for NUTS. As it is cleared that mismatch of sequence number will results in denial of service. Therefore, it could be first tested the reliability of commands to synchronize SN before launching.

We have proposed that time stamp could be a good and strong solution against replay attack. This could be studied further. Nevertheless, first the decision should be made by NUTS authority about the implementation of relative or absolute time in CubeSat.

It is found that GENSO network is suitable for CubeSat where ground station is idle for around 90% of time. IN this scenario concept of joining GENSO sounds interesting. However, access control and authentication in the GENSO network can be analyzed.

Finally, the main task remaining will be to develop the software implementing all the proposed methodology. We are already lagging in software in compared to hardware.

# REFERENCES


[1] A. Toorian, E. Blundell, J. P. Suari and R. Twiggs, "CubeSats as responsive satellites," *Aerospace Engineering,* vol. 805, pp. 756--6479, 2005.

[2] K. Cote, "Mechanical, Power, and Propulsion Subsystem Design for a CubeSat," *WORCESTER POLYTECHNIC INSTITUTE,* 2011.

[3] H. Heidt, J. Puig-Suari, A. Moore, S. Nakasuka and R. Twiggs, "CubeSat: A new generation of picosatellite for education and industry low-cost space experimentation," *Proceedings of the 14th Annual AIAA/USU Conference on Small Satellites,* pp. 1--19, 2000.

[4] R. Birkeland, "NTNU Test Satellite NUTS-1 Mission Statement," 2011.

[5] Wikipedia, "AVR32", http://en.wikipedia.org/wiki/AVR32, Accessed date 27/022012.

[6] Atmel,"Datasheet",*http://www.atmel.com/Images/doc7919.pdf,* Accessed date 27/02/2012.

[7] K. Andersson and R. Andersson, "A comparison between FreeRTOS and RTLinux in embedded real-time systems," *Linköping University,* Suecia, 2005.

[8] W. Stallings, Cryptography and network security, Prentice hall, 2003.

[9] H. Krawczyk, R. Canetti and M. Bellare, "HMAC: Keyed-hashing for message authentication,", *ITEF*, 1997.

[10] AVR, "Atmel AVR32931: UC3-L0 Xplained Getting Started Guide," *http://www.atmel.com/Images/as5installer-stable-5.1.208-readme.pdf,* Accessed date 21/04/2012.

[11] Atmel, "Datasheet," *http://www.atmel.com/devices/at32uc3a3256.aspx#datasheets,* Accessed date 21/04/2012.

[12] GomSpac, "CSP library," *https://github.com/GomSpace/libcsp,* Accessed date 02/03/2012.

[13] Wikipedia, "CubeSat Space Protocol," *http://en.wikipedia.org/wiki/Cubesat_Space_Protocol* , Accessed date 21/04/2012.

[14] J. Kelsey, B. Schneier and D. Wagner, "Related-key cryptanalysis of 3-way, biham-des,

cast, des-x, newdes, $RC_2$, and tea," *Information and Communications Security*, pp. 233--246, 1997.

[15] J. Lu, "Related-key rectangle attack on 36 rounds of the XTEA block cipher," *International Journal of Information Security*, vol. 8, no. 1, pp. 1--11, 2009.

[16] G. Sekar, N. Mouha, V. Velichkov and B. Preneel, "Meet-in-the-middle attacks on reduced-round XTEA," *Topics in Cryptology--CT-RSA 2011*, pp. 250--267, 2011.

[17] V. Visockas, "Access control and securing of the NUTS uplink," NTNU, 2011.

[18] J. P. Kaps, "Chai-tea, cryptographic hardware implementations of xTEA," *Progress in Cryptology-INDOCRYPT 2008*, pp. 363--375, 2008.

[19] G. Gaubatz, J. Kaps and B. Sunar, "Public key cryptography in sensor networks—revisited," *Security in Ad-hoc and Sensor Networks*, pp. 2--18, 2005.

[20] X. Wang, Y. Yin and H. Yu, "Finding collisions in the full SHA-1," in *Advances in Cryptology--CRYPTO 2005*, Springer, 2005, pp. 17-36.

[21] L. G. Roberts, "Beyond Moore's law: Internet growth trends," *Computer*, vol. 33, no. 1, pp. 117--119, 2000.

[22] D. Gligoroski, V. Klima, S. J. Knapskog, M. El-Hadedy, J. Amundsen and S. f. Mjølsnes, "Cryptographic Hash Function BLUE MIDNIGHT WISH," *NTNU*, Trondheim, 2008.

[23] D. Gligoroski, L. Kocarev, R. S. Ødegård, M. Mihova and S. J. Knaspog, "Cryptographic Hash Function EDON-R," *NTNU*, Trondheim, 2008.

[24] L. Andreicheva, "Attacks on SHA-3 candidate functions: Keccak and Blue Midnight Wish (BMW)," 2011.

[25] J. Edney and W. A. Arbaugh, Real 802.11 security: Wi-Fi protected access and 802.11 i, *Addison-Wesley Professional*, 2004.

[26] W. A. Beech, D. E. Nielsen and J. Taylor, "AX. 25 Link Access Protocol for Amateur Packet Radio," *Tucson Amateur Packet Radio Corporation web site, version*, vol. 2, 1998.

[27] P. Oechslin, "Making a faster cryptanalytic time-memory trade-off," *Advances in Cryptology-CRYPTO 2003*, pp. 617--630, 2003.

[28] F. George , "Telemetry and Telecommand Transfer Frames Format," *SwissCubesat*, 2007.

[29] R. D. Standard and R. Book, "CCSDS cryptographic algorithms," 2010.

[30] AAUSAT II,"Alborg University CubeSat ", *http://www.space.aau.dk/aausatii/eng/index.php?n=Site.GndSetup, Accessed date* 18/05/2012.

[31] AAUSAT II, "Alborg University CubeSat Protocol", *http://www.space.aau.dk/aausatii/homepage/index.php?language=en&page=dok/hsn,*" Accessed date 18/05/2012.

[32] AubieSat, "Alburn University CubeSat", *http://space.auburn.edu/index.htm,* Accessed date 18/05/2012.

[33] AMSAT-UK, "Armateur Radio CubeSats Launch", *http://www.uk.amsat.org/2271*," Accessed date 18/05/2012.

[34] ECRYPT Benchmarking of Cryptographic Systems, "eXternal Benchmarking eXtension", *http://bench.cr.yp.to/xbx.html*, Accessed date 23/05/201.

[35] ECRYPT Benchmarking of Cryptographic Systems, "Measurements of hash functions", *http://bench.cr.yp.to/results-hash.html*, Accessed date 23/05/2012.

[36] M. Bellare, "New proofs for NMAC and HMAC: Security without collision-resistance," *Advances in Cryptology-CRYPTO 2006*, pp. 602--619, 2006.

[37] C. Madson and R. Glenn, "The use of HMAC-MD5-96 within ESP and AH," RFC, 1998.

[38] ECRYPT Benchmarking of Cryptographic Systems, "SUPERCOP", *http://bench.cr.yp.to/supercop.html*, Accessed date 24/05/2012.

[39] CCSDS, "CCSDS Key Management Techniques," *CCSDS Secretariat*, Washington DC, 2005.

[40] GENSO, "Introduction of GENSO", *http://www.genso.org*, Accessed date 13/06/2012.

# APPENDIX A

## Proposed Telemetry signals for NUTS

Note: These signals are proposed by NUTS member working in communication protocol.

Signal classes. (Maybe we could use only 8 bits for this. We'll have to revise)

- Backplane voltages: 0x0100
- Backplane currents: 0x0200
- Temperatures: 0x0300
- EPS: 0x0400
- ADCS: 0x0500
- Payload: 0x0600
- OBC: 0x0700
- Radio: 0x0800

| ID | Signal | Description | Size | Resolution | Range | Value |
|---|---|---|---|---|---|---|
| 0x0101 | MOD1_3V_V | 3 V bus voltage | 16 | 0.01 V | 0 - 5.12 V | meas. value * scale = phys. value |
| 0x0201 | MOD1_3V_A | 3 V bus current | 16 | 2 mA | 0 - 1024 mA | |
| 0x0102 | MOD2_3V_V | 3 V bus voltage | | | | |
| 0x0202 | MOD2_3V_A | 3 V bus current | | | | |
| 0x0103 | MOD3_3V_V | 3 V bus voltage | | | | |
| 0x0203 | MOD3_3V_A | 3 V bus current | | | | |
| 0x0104 | MOD4_3V_V | 3 V bus voltage | | | | |
| 0x0204 | MOD4_3V_A | 3 V bus | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | current | | | | |
| 0x0105 | MOD5_3V_V | 3 V bus voltage | | | | |
| 0x0205 | MOD5_3V_A | 3 V bus current | | | | |
| 0x0106 | MOD6_3V_V | 3 V bus voltage | | | | |
| 0x0206 | MOD6_3V_A | 3 V bus current | | | | |
| 0x0107 | MOD7_3V_V | 3 V bus voltage | | | | |
| 0x0207 | MOD7_3V_A | 3 V bus current | | | | |
| 0x0108 | MOD8_3V_V | 3 V bus voltage | | | | |
| 0x0208 | MOD8_3V_A | 3 V bus current | | | | |
| 0x0109 | MOD1_5V_V | 5 V bus voltage | 16 | 0.02 V | 0 - 10.24 V | |
| 0x0209 | MOD1_5V_A | 5 A bus current | 16 | 2 mA | 0 - 1024 mA | |
| 0x010A | MOD2_5V_V | 5 V bus voltage | | | | |
| 0x020A | MOD2_5V_A | 5 A bus current | | | | |
| 0x010B | MOD3_5V_V | 5 V bus voltage | | | | |
| 0x020B | MOD3_5V_A | 5 A bus current | | | | |
| 0x010C | MOD4_5V_V | 5 V bus voltage | | | | |
| 0x020C | MOD4_5V_A | 5 A bus current | | | | |
| 0x010D | MOD5_5V_V | 5 V bus | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | voltage | | | | |
| 0x020D | MOD5_5V_A | 5 A bus current | | | | |
| 0x010E | MOD6_5V_V | 5 V bus voltage | | | | |
| 0x020E | MOD6_5V_A | 5 A bus current | | | | |
| 0x010F | MOD7_5V_V | 5 V bus voltage | | | | |
| 0x020F | MOD7_5V_A | 5 A bus current | | | | |
| 0x0110 | MOD8_5V_V | 5 V bus voltage | | | | |
| 0x0210 | MOD8_5V_A | 5 A bus current | | | | |
| | | | | | | |
| 0x0301 | BATT_TEMP | Battery temp | | | | |
| 0x0302 | RADIO_TEMP | Radio PCB temp | | | | |
| 0x0303 | ZENIT_TEMP | Zenit panel temp | | | | |
| 0x0304 | NADIR_TEMP | Nadir panel temp | | | | |
| | | | | | | |
| 0x0401 | CELL_A_V | Solar cell A voltage | 16 | 0.01 V | 0 - 5.12 V | |
| 0x0402 | CELL_A_A | Solar cell A current | 16 | 1.2 mA | 0 - 614.4 mA | |
| 0x0403 | CELL_B_V | Solar cell B voltage | | | | |
| 0x0404 | CELL_B_A | Solar cell B current | | | | |
| 0x0405 | CELL_C_V | Solar cell C | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | voltage | | | | |
| 0x0406 | CELL_C_A | Solar cell C current | | | | |
| 0x0407 | CELL_D_V | Solar cell D voltage | | | | |
| 0x0408 | CELL_D_A | Solar cell D current | | | | |
| 0x0409 | CELL_E_V | Solar cell E voltage | | | | |
| 0x040A | CELL_E_A | Solar cell E current | | | | |
| | | | | | | |
| 0x0701 | OBC_RTC_EPOC | Seconds since epoc | | | | seconds since epoc |
| | | | | | | |
| 0x0501 | ADCS_MAG_X | Magnetometer X value [nT] | 16 | TBD | TBD | |
| 0x0501 | ADCS_MAG_Y | Magnetometer Y value [nT] | | | | |
| 0x0501 | ADCS_MAG_Z | Magnetometer Z value [nT] | | | | |
| 0x0501 | ADCS_SAT_DIR | Pointing direction | | TBD | TBD | TBD |
| 0x0501 | ADCS_MODE | Regulated or “free” | | | | |
| | | | | | | |
| 0x0410 | MOD1_V_STAT | Status of voltage bus | 2 | | | ON/OFF/ERR |
| 0x0411 | MOD2_V_STAT | | | | | |
| 0x0412 | MOD3_V_ST | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | AT | | | | | |
| 0x0413 | MOD4_V_STAT | | | | | |
| 0x0414 | MOD5_V_STAT | | | | | |
| 0x0415 | MOD6_V_STAT | | | | | |
| 0x0416 | MOD7_V_STAT | | | | | |
| 0x0417 | MOD8_V_STAT | | | | | |
| | | | | | | |
| 0x0710 | MOD1_BUS_STAT | Status of I2C bus | 2 | | | ON/OFF/ERR |
| 0x0712 | MOD2_BUS_STAT | | | | | |
| 0x0713 | MOD3_BUS_STAT | | | | | |
| 0x0714 | MOD4_BUS_STAT | | | | | |
| 0x0715 | MOD5_BUS_STAT | | | | | |
| 0x0716 | MOD6_BUS_STAT | | | | | |
| 0x0717 | MOD7_BUS_STAT | | | | | |
| 0x0718 | MOD8_BUS_STAT | | | | | |
| 0x0420 | EPS_MODE | | 2 | | | SIMPLE/MPPT TBD |
| 0x0421 | EPS_5V_A_STAT | | 2 | | | ON/OFF/ERR (Can be determined???) |

| | | | | | | |
|---|---|---|---|---|---|---|
| 0x0422 | EPS_5V_B_STAT | | | | | |
| 0x0423 | EPS_3V_A_STAT | | | | | |
| 0x0424 | EPS_3V_B_STAT | | | | | |
| | | | | | | |
| | EPS_ | | | | | |
| | | | | | | |
| 0x0720 | OBC_CPU_LOAD | CPU-load | 8 | 0.4 % | 0 - 100 % | |
| 0x0721 | OBC_CPU_FREQ | | 8 | 0.4 Hz | 0 - 100 Hz | Needed???? |
| 0x0722 | OBC_FLASH_USE | % of used flash space | 8 | 0.4% | 0 - 100 % | |
| 0x0723 | OBC_IMG_VER | Version of running image | 8 | | | Needed???? |
| | | | | | | |
| 0x0801 | COMM_VHF_STAT | Status of VHF-radio | 8 | | | May be wise to make a bit-mask and extract data from this. RX/TX/LOW_PWR/ERR osv.... |
| 0x0802 | COMM_UHF_STAT | Status of UHF-radio | 8 | | | As above |
| 0x0803 | COMM_B_STAT | Beacon status | 8 | | | ON/OFF/CHANGED/ORIGINAL …. TBD |
| | | | | | | |

## Command set format

Command set proposed to NUTS by NUTS member working in communication protocol.

## Request format

| 16 | 48 |
|---|---|
| **Command type** | **Command argument** |

| **Request type** | **Description** | **Parameter** | **Response** |
|---|---|---|---|
| CAM_TAKE_PIC | take picture | N/A | |
| CAM_SET_EXP | exposure time | exposure time in millisecnds | |
| CAM_SET_RES | resolution | | |
| | | | |
| OBC_TAKE_PIC | Schedule CAM_TAKE_PIC | When the CAM_TAKE_PIC should be issued to the camera, in seconds relative to now | |
| GEN_GET_STATUS | Get sensor status | N/A | |

| | | | |
|---|---|---|---|
| ADCS_DETUMBLE | | | |
| ADCS_EARTH_ORIENT | | | |
| ADCS_ROT_Z | | Target radians/sec | |
| ADCS_ROT_Z | | Target radians/sec | |
| ADCS_ROT_Y | | Target radians/sec | |
| GEN_SYSTEM_RESET | | | |
| GEN_RESET_SUBSYS | | Module number | |
| GEN_PWR_ON_SUBSYS | | Module number | |
| GEN_PWR_OFF_SUBSYS | | Module number | |
| RADIO_BEACON_ON | | N/A | |
| RADIO_BEACON_OFF | | N/A | |
| RADIO_BEACON_START | | | |
| RADIO_BEACON_APPEND | | | |
| RADIO_BEACON_END | | | |
| EPS_GET_STATUS | | | |
| EPS_SET_MODE | | | |

**Response format**

On successful reception and handling of a request, the success bit will be set high.

| | | | | | |
|---|---|---|---|---|---|
| 1 | 7 | 1 | 7 | 24 | 2^24 |

| Success bit | Command type | Last response | Options | Length | Payload |
|---|---|---|---|---|---|

| Response type | Description | Parameter | Parameters |
|---|---|---|---|
| RES_GEN_ERROR | General error | N/A | |
| RES_CAM_TAKE_PIC | Response to take a picture command | | |