CubeSec and GndSec: a lightweight security solution for CubeSat communications

Obulapathi N Challa, Gokul Bhat, Dr. Janise Mcnair
University of Florida
Department of Electrical and Computer Engineering
P.O. Box 116130
Gainesville, FL 32611 USA
352-234-6767
obulpathi@ufl.edu

ABSTRACT

Communication protocols implemented for CubeSat networks have trivial overhead and almost no security features. As CubeSats are heavily constrained for resources, complex security suites and protocols can seldom be implemented. Cyclic Redundancy Check (CRC) which is currently used in CubeSats provides no protection against intentional corruption of data and moreover, CubeSats are vulnerable to eavesdropping due to the wireless channel. Message integrity also becomes questionable as an attacker can modify commands and data. This paper proposes CubeSec and GndSec, a very light-weight security solution for CubeSats communications. CubeSec and GndSec provides mutual authentication, confidentiality, data integrity between Cubesat and ground Station using preshared keys.

INTRODUCTION

With the introduction of CubeSat networks like "Massive Operations, Recording, and Experimentation Database System" (MoreDB) [1] by Cal Poly, "Global Educational Network for Satellite Operations" (GENSO) [2] by European Space Agency (ESA), data and commands can be sent to CubeSat from multiple ground stations. Communication protocols implemented for CubeSat networks have trivial overhead and almost no security features. Since, CubeSats use wireless broadcast media, they are susceptible to eavesdropping and any unauthorized user can intercept and monitor the sequence of commands sent to and received from CubeSat. After gathering sufficient amount of data and analyzing it, the adversary can perform replay attacks on the Cubesat. Also, the attacker can send spurious commands causing excessive resource consumption, data loss or corruption on satellite and eventually mission failure.

Given that a CubeSat can typically generate 2W power and has a constrained 8 minutes window time per pass (with an average exposure time of 25 minutes per day), security mechanisms like authentication and encryption increase latency, power consumption and reduce the network efficiency. Currently, CubeSats use Cyclic Redundancy Check (CRC) for checking correctness of received data. CRC provides a good defense against accidental data transmission errors, but provides no protection against intentional corruption of data. Thus, in order to strengthen the resilience to security attacks on CubeSat Networks, the problem of security must be addressed. This paper presents a novel lightweight CubeSat security subsystem called "CubeSec" which is a low cost (100$), small form factor (5cm x 5cm), low power (10mW) and low weight (10 gms) security solution for CubeSat communication. The following section presents an overview of CubeSat security and presents the challenges for CubeSat security subsystem. Section III describes the block ciphers, various modes of encryption and their comparison. Section IV provides details about how CubeSec is implemented. Section V presents Size, Weight and Power (SWaP) analysis of the CubeSec subsystem and conclusion along with the future work is provided in section VII.

OVERVIEW OF CUBESAT SECURITY

Security is an important topic in the age of information era. With recent super advanced security threats like Stuxnet, Flame, leak of password database on LinkedIn.com and last.fm securing computer systems is more important than ever. Attacks like this are signalling the dawn of a new era in cyber security warfare. Stuxnet worm, discovered in June 2010, spread from one computer to another and caused issues and infected Siemens industrial equipment and software that was running on a Windows system. Stuxnet targeted the uranium enrichment infrastructure in Iran. Similar to Stuxnet, Flame was another very advanced malware that used chosen-prefix collision attack to to digitally sign malicious Flame components.

Things get even more complicated in case of systems like CubeSats, where there is no direct access to the physical system once its launched and is in space. And with introduction of CubeSat networks like GENSO, the trouble can arise from anywhere in the system. We need strong security measures in order to strengthen the resilience of CubeSats against security attacks.

Components of Data Security

Data security consists of three components:

1. Confidentiality
2. Integrity
3. Availability

which together are referred to as CIA. Here is a brief description of them.

Confidentiality: Confidentiality refers to the property that data can only be read by the authorized parties. Even if data is transmitted correctly over network ensuring its integrity, if any machine or person accesses the data without permission, it is no longer confidential. Confidentiality brings up questions like who has access to data and which components of it and how is it protected from unauthorized from access.

Integrity: Data integrity refers to the property data is valid and has not been tampered with. Integrity comes into play when either data is stored or exchanged. Received data must be same as the transmitted data.

Availability: Availability refers to the property that data is available when requested. Integrity and confidentiality are useless if data cannot be accessed and is not available. Denial-of-Service (DoS) attacks target availability of service and make service unavailable or unusable by consuming all available network, CPU, storage, or system resources.

Challenges for CubeSat security subsystem

Power constraints: CubeSat has limited power generation capability of about 2W. So CubeSat security subsystem has to efficient in terms of power and communication overhead.

Space constraints: CubeSat volume is limited to 10 x 10 x 10 cm cubed. As a result CubeSat security subsystem has to be space efficient.

Weight constraints: Weight of CubeSat is constrained to 1000 gms. So CubeSat security subsystem should be weight efficient.

Time constraints: CubeSat has a constrained 8 minutes window time per pass with an average exposure time of about 25 minutes per day. So security protocol for CubeSats should not incur significant latency which will increase power consumption and reduce the bandwidth. Furthermore, higher latency reduces Quality of Service (QoS) for voice and other real time applications. In addition to the above requirements, the subsystem needs redundancy in order to mitigate the risk of its failure.

BLOCK CIPHERS AND MODES OF ENCRYPTION

Block ciphers are fundamental building components in the design of cryptographic protocols. Block cipher can be treated as a deterministic function that transforms a block of bits into its corresponding ciphertext. This mapping is determined by the symmetric key. AES and DES are are the most prominent block ciphers. Here is an overview of AES and DES.

Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES), also known as Rijndael, was developed by J. Daemen and V. Rijmen. It was announced as a security standard by National Institute of Standards and Technology (NIST) in 2001. AES is the de-facto industry standard for symmetric block cipher and is widely deployed. AES has a fixed block size of 128 bits and variable key size of 128, 192, or 256 bits. It operates on a 4×4 byte matrix. The AES uses 10, 12 and 14 cycles of repetitions of transformation rounds for 128, 192 and 256 bit keys to convert the input plaintext into the final output of ciphertext. Each round consists of four steps, namely AddRoundKey, SubBytes, ShiftRows, and MixColumns. Owing to its byte-oriented design, AES is very efficient on low end microcontrollers.

Data Encryption Standard (DES)

Data Encryption Standard (DES) was made as standard by Federal Information Processing Standard (FIPS) for the United States in 1976. The block size in DES is 64 bits. The cipher key size is 56 bits. However, only 56 of these are actually used by the algorithm, the other ones are parity check bits. Core of DES is the Feistel network containing 16 rounds with 8 substitution boxes along with an initial permutation, a final permutation and a key schedule. DES was proved to be insecure by DES crackers like EFF DES Cracker [3] or the COPA-COBANA [4]. However DES3 is still used.

For encrypting a small block of data, AES or DES will suffice. However for encrypting a large amount of data we need to encrypt many blocks, each one using AES or DES block cipher. this can be achieved by chaining

the individual encryptions. Electronic codebook (ECB), Cipher Block Chaining (CBC), Propagating Cipher Block Chaining (PCBC), Cipher Feedback (CFB), Output Feedback (OFB), Counter mode (CTR), Galois/Counter Mode of Operation (GCM) are the most prominent chaining modes of operation. Here we present their characteristics and compare these modes of encryption.

Electronic codebook (ECB)

Electronic codebook (ECB) is a very simple mode of encryption which divides the data into fixed size blocks and encrypts each block separately.

There are several problems of ECB. One of the major problems of ECB is that it encrypts deterministically. Same data blocks are encrypted to same ciphertext. As a result, it does not hide patterns and thus not recommended for use in cryptographic protocols. Protocols without integrity protection are susceptible to replay attacks with ECB, since each block gets decrypted in exactly the same way.

Cipher-block chaining (CBC)

Cipher-block chaining (CBC) mode of operation was invented by IBM. In CBC, each block of plaintext is XORed with ciphertext corresponding to previous block. As a result, identical data blocks are encrypted to different ciphertext blocks. Each ciphertext block depends on all the previous plaintext blocks. An initialization vector is employed to in order to generate different ciphertext for same message, each time it is encrypted.

Despite being one of the most commonly used mode of operation, CBC can drawbacks including, sequential encryption. Multiple blocks cannot be encrypted in parallel as the result on one block depends on the output of all the previous block till that point. CBC also requires padding of message to a multiple of the cipher block size. One-bit change to the ciphertext causes complete corruption of the corresponding block of plaintext, and inverts the corresponding bit in the following block of plaintext, but the rest of the blocks remain intact.

Propagating cipher-block chaining (PCBC)

Propagating cipher-block chaining or plaintext cipher-block chaining mode was designed to cause small changes in the ciphertext to propagate indefinitely when decrypting, as well as when encrypting.

PCBC also suffers from the problem that messages cannot be encrypted in parallel. Also, if two adjacent ciphertext blocks are exchanged, this does not affect the decryption of subsequent blocks.

Cipher feedback mode(CFB)

Cipher feedback (CFB) mode is similar to CBC. CFB generates key stream using previous cipher block as IV. CFB can be used as a self-synchronizing stream cipher that will synchronize for any multiple of x bits lost.

Compared to OFB and CTR, CFB has several advantages including that the block cipher is only ever used in the encrypting direction and the message does not need to be padded to a multiple of the cipher block size.

Output feedback mode(OFB)

Output feedback (OFB) is a synchronous stream cipher. Keystream blocks are generated and are XORed with the plaintext blocks to get the ciphertext. Similar to other stream ciphers, flipping a bit in the ciphertext flips the bit in plaintext in corresponding location. Owing to its symmetrical XOR operation, encryption and decryption are exactly the same.

Even though encryption of different messages cannot be performed in parallel, since output feedback block cipher operation depends on all previous ones, all block cipher operations can be performed in advance, so that XORing of different blocks can be done in parallel.

Counter mode(CTR)

CTR mode (CM) is also a stream cipher. Ciphertext is obtained by XORing the encrypted successive values of counter with corresponding data blocks. CTR mode is one of the most widely used ciphers. CTR mode is very similar to OFB mode. CTR mode is well suited for parallel encryption of message blocks using multiple processors.

Galois/Counter Mode (GCM)

Galois/Counter Mode (GCM) mode of operation is widely adopted for its efficiency and performance. It uses universal hashing over a binary Galois field to provide authenticated encryption. It can be implemented with reasonable hardware resources to achieve very high speed of encryption. For software implementations, table driven field operations can be employed to achieve excellent performance. GMAC is an authentication-only variant of the GCM which can be used as an incremental message authentication code. Both GCM and GMAC can accept initialization vectors of arbitrary length [5].

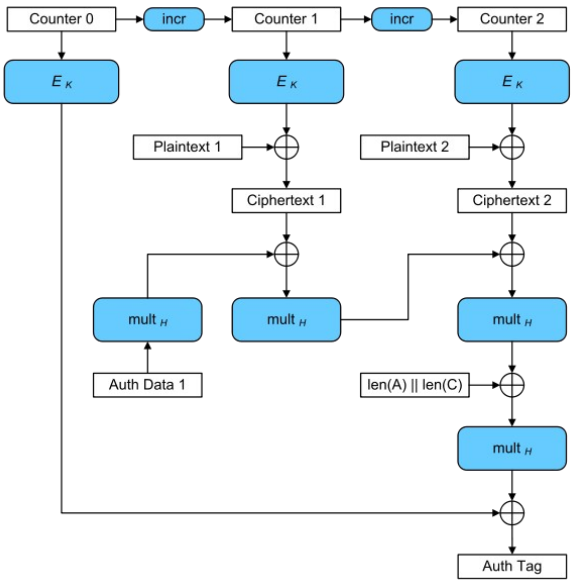

Figure 1: GCM Encryption Operation

Comparison of ECB, CBC, PCBC, CFB, OFB and GCM

We compared ECB, CBC, PCBC,CFB, OFB and GCM modes of operation for use of encrypting data on CubeSec. ECB and CBC require padding of final block with zeroes to round it to the block size of 128 bits. With 256 byte sized packets (AX.25) and 16 bytes block size, this means an overhead of 3 % in best case. With smaller sized packets, used for control information, this overhead can be as much as 25%. CBC, PCBC, CFB and OFB modes does not support parallelism as the input for encrypting or decrypting a block depends on the result of the encryption or decryption of the previous block. GCM does not require padding of zeroes for final block. GCM allows pipelined and parallelized implementations and have minimal computational latency to achieve high encryption speed. Parallel processing enables use of additional microcontrollers, provided for hardware redundancy, for increasing encryption speed. GCM is also free from intellectual property rights.

GCM has other additional features like being capable of acting as a stand-alone MAC, authenticating messages when there is no data to encrypt, with no modifications. It can be used for incremental message authentication. Owing to its speed, ability to parallelize, cost of implementation and other above mentioned advantages, we selected GCM mode for CubeSat security subsystem [5].

IMPLEMENTATION

As a proof of concept, symmetric encryption using AES and DES with GCM mode of operation is implemented using ATXMega128 microcontroller on XPlain evaluation kit from Atmel. Here is a detailed explanation of why we choose hardware implementation of AES and DES block ciphers over software implementation, why we selected ATXMega128 as our choice of microcontroller and XPlain as our platform to implement the security solution for CubeSats.

Software vs Hardware implementation

Software implementation of AES requires 3766 CPU cycles for encryption and 4558 CPU cycles for decryption per block. Software implementation of DES requires 8633 CPU cycles for encryption and 8154 CPU cycles for decryption per block [6]. Whereas hardware implementation of AES and DES require about few 100's of CPU cycles for encryption or decryption on a typical microcontroller [6]. This means hardware implementation of AES and DES is about 25 times efficient in terms of time and power compared to software implementation. So we wanted to implement AES and DES using hardware. We surveyed Atmel and PIC microcontrollers featuring hardware implementation of AES and DES.

XMega family of microcontrollers from Atmel were best suited for CubeSec. AES or DES requires only 375 cycles for encryption and decryption. Direct Memory Access (DMA) enabled AES and DES crypto-engines facilitates fast encryption and data transfer between memory, crypto-engines and other IO peripherals to boost performance [7]. Here is an overview of XMega family of microcontrollers.

Overview of XMega family of microcontrollers

XMega devices are based on 8/16-bit AVR RISC core with 135 RISC instructions. XMegas include features like AES and DES crypto, flexible Timer/Counters, Direct Memory Access, multiple communication modules and types, and efficient power management. The DMA controller, innovative event system ensure predictable real time performance even with high load. AVR XMEGA operates from just 1.6 volt with up to 32 MIPS at 32 MHz. Memory size ranges from 16 to 384 KB of Flash, 8 KB EEPROM and 8 kB SRAM and they come in 44- to 100-pin packages [7]. The highly integrated design minimizes bill of materials in a broad range of applications. To speed up development, all AVR XMEGA devices are code-compatible. Code can be reused across all XMega devices.

Key features of XMega family of microcontrollers:

1. Ultra Low Power: Atmel's picoPower technology is uses very low power. True 1.6 volt operation enables battery powered applications with XMega devices to be connected to a 1.8V regulated power supply to save cost and battery life. XMegas maintain RTC and full SRAM retention using only 550 nA.

2. DMA Controller: XMEGA features a multi-channel DMA controller that enables fast, CPU-independent data transfer between any combination of data memories and peripherals. Ultra low power consumption coupled with DMA for data transfer with out processor interruption make the application consume very low power. CPU-independent data transfer using DMA between AES, 8 kB of RAM and hardware multiplier make the process of collecting, storing and processing the data faster that significantly boosts performance.

3. Fast inter-peripheral signaling: The innovative event system enables inter-peripheral signaling without CPU or DMA usage, for short and guaranteed response time.

4. Software Library: All XMega devices are supported by AVR Software Framework, a complete library of device drivers and middleware.

5. Hardware Multiplier: Hardware multiplier is one module of very interest for the Security and Signal Processing Applications. Multiplier is capable of multiplying two 8-bit numbers into a 16-bit result. The hardware multiplier supports different variations of signed and unsigned integer and fractional numbers. Multiplication takes only two CPU clock cycles

Among the XMega controllers, ATXMega128 Microcontroller was chosen for implementing the CubeSec. It has 128 KB of Flash, 8 KB EEPROM and 8 kB SRAM [8].

Hardware platform

The hardware platform we choose to use for this project is the Xplain board from Atmel AVR. The Xplain evaluation kit is a hardware platform to evaluate the ATXMega128 microcontroller. XPlain enables testing the peripherals of XMEGA peripherals right away to get an understanding of how to use peripherals of the XMEGA. XPlain comes with several features like Integrated USB Communication Gateway, JTAG Programming headers, attached flash memory [9].

Figure 2: Xplain Hardware Platform

Here is a detailed explanation of key components of XPlain board:

1. ATXMega128: The core of the XPlain board is the ATXMega128 microcontroller, whose JTAG, Analog and Digital IO, SPI, I2C, USART Ports are connected to several peripherals like JTAG, Analog and Digital IO Headers, Flash Memory, USB Gateway. ATXMega128 can be programmed through JTAG Header.

2. Power Supply: The kit is powered from USB which leaves two options to power it: Either connect the kit to a PC through an USB cable, or to a 5V USB power supply (AC/DC adapter).

3. Communication through UART-to-USB gateway: The XMEGA's UARTC0 is connected to a software UART on the AT90USB1287. When the AT90USB1287 device is enumerated (connected to a PC) the data transmitted from the XMEGA is passed on to a (virtual) COM port.

4. Attached Memories: The Xplain kit demonstrates how to use the External Bus Interface (EBI) module to interface a 4-bit SDRAM. An 8MB SDRAM (16M x 4) is attached in 3-port EBI mode (PORTH, PORTK and PORTJ). An 8MB serial Dataflash is connected to the UARTC1 which can operate in SPI master mode to interface the dataflash.

5. IO expansion headers: The XMEGA's analog PORTA is available on the "XMEGA PORT A" pin header (J101). This allows the user to connect external signals to the ADC, DAC and Analog Comparators on PORTA.

6. Programming headers: In addition to programming the XMEGA through the AT90USB1287, the XMEGA can be programmed and debugged by connecting an external programming/debugging tool to the "JTAG & PDI XMEGA" pin header (J100). The pin header is having a standard JTAG programmer pinout (refer to online help in AVR Studio), and tools like the JTAGICE mkII can thus be connected directly to the header. If it is desired to use PDI programming/debugging an adapter must be used.

7. The XMEGA's digital PORTD is available on the "XMEGA PORT D" pin header (J102). This port features general purpose IO and various communication modules (USART, SPI and TWI).

8. Miscellaneous IO: Eight micro switch buttons are connected to the XMEGA's PORTF. Internal pull-ups should be enabled to detect when the buttons are pushed as they short the respective line to GND. Eight LEDs are connected to the XMEGA's PORTE. The LEDs are active low – and thus lights up when the respective lines are drawn low by the XMEGA.

Software: Installation and Configuration.

The code was developed on Ubuntu GNU/Linux system [10]. For developing the code, GNU toolchain had been used [11. The GNU C-development toolchain consists of compiler, binary utilities and C-library. All the tools are available for Atmel's AVR target.

1. gcc-avr Gnu C Compiler for AVR

2. binutils-avr Binary utilities (linker, assembler, etc.) for AVR

3. avr-libc Basic C-library for AVR

4. gdb-avr Gnu debugger for AVR

All the required software can be installed on a Debian based platform (Debian and Debian derivatives: Knoppix, Ubuntu, SELinux, etc.) using apt tool. There exists ready made installation packages in standard sources servers for GNU development tools. To install these packages, use the following command: "$ sudo apt-get install gcc-avr binutils-avr avr-libc gdb-avr". This will install all the necessary development tools: gcc-avr, binutils-avr, avr-libc and gdb-avr.

Programming the hardware

Avrdude [12] software is a programming tool used to program AVR microcontrollers. Avrdude supports wide range of programming tools and target AVR devices. It runs on Linux, FreeBSD, Mac OSX and Win32. Projects home page: http://savannah.nongnu.org/projects/avrdude. Avrdude's configuration file allows user to define alternative programmer cable pin configurations. Thus it can be used with exotic programming cables as well. To install Avrdude, use the following command: "$sudo apt-get install avrdude". Avrdude can be used with variety of programming hardware like AVRISP mkII, JtagICE mkII etc .,

CUBESEC AND GNDSEC

Implementing AES and DES block cipher in software on CubeSat is resource intensive and time consuming. So decided to use microcontrollers with AES and DES hardware support. After filtering through various microcontrollers from Atmel and PIC, we selected ATXMega128 from XMega microcontroller family as our final candidate for implementing CubeSec. ATXMega128 microcontroller has Direct Memory Access (DMA) enabled AES crypto-engine. This facilitates fast encryption and data transfer between memory and peripherals to boost performance.

GCM has several advantages over other modes of encryption. They include pipelined and parallelized implementation, minimal computational latency to achieve high encryption speed, ability to use redundant microcontrollers, ability to act as a stand alone MAC and incremental message authentication. Owing to its speed, ability to parallelize, cost of implementation and other above mentioned advantages, we selected GCM mode for CubeSec.

As a proof of concept, symmetric key encryption using 128 bit Advanced Encryption Standard (AES) and Data Encryption Standard (DES) block encryption algorithms in Galois/Counter Mode (GCM) of operation has been implemented on XPlain platform hosting the ATXMega128 microcontroller. CubeSec software is available for download from GitHub [13].

GndSec is the terrestrial counterpart of CubeSec. We implemented required encryption software for GndSec in python software. CubeSec and GndSec provide

mutual authentication, confidentiality, data integrity between CubeSat and ground station.

SIZE, WEIGHT AND POWER (SWAP) ANALYSIS

Size Analysis

ATXMega128 microcontroller which forms the core of the application is 2cm x 2 cm. Other than power connections, it also needs JTAG programmer for programming and debugging purposes. JTAG can also be used for configuring the AVE ECU with encryption keys and any other configuration data. Including the, standard mini JTAG connector (10 with 1mm spacing between pins, dual row) with has size of 6mm x 3mm, total footprint is below 3cm x 3cm. Size of CubeSec with two microcontrollers, to provide redundancy and increase speed of encryption size, is less than 5cm x 5 cm.

Weight Analysis

Weight of PCB is about 0.2 gm/cm2. 5 cm x 5 cm PCB weighs about 5gms. ATXMega128 microcontroller and power circuitry weighs about 2 gms, mini JTAG connector weight about 1.5 gms. Including the weight of soldering and other miscellaneous components, which add up to about 1.5 gms, total weight of the system is about 9.6 gms.

Power Analysis

ATXMega128 in active mode operating at 3.0 V with external clock running at 1 MHz drains 800uA. As a result, 2.4 mW of power consumed at 1 MHz Operation.

Data Rate

Encrypting a single 16 byte AES block of data requires 375 clock cycles. At a clock rate of 1MHz, data stream can be encrypted at the rate of 43KBps using AES block encryption algorithm. In Galois/Counter Mode (GCM) mode of operation data stream can be encrypted at 5.4KBps. However, this speed can be scaled by using higher clock rate. Using 25 mW of power, ATXMega128 can be run at 12 MHz which gives encryption rate of 64 KBps. Using 100 mW of power and two microcontrollers, encryption speed can be increased to 256 Kbps.

Summary of SWaP analysis

CubeSec can be constructed with minimum requirements: size of 3 cm x 3 cm, weighing 10 grams and power requirement of 2.5 mW to encrypt data stream at a rate of 5.4 KBps. Data encryption speed can be scaled up to 256KBps by increasing the power consumption to 100 mW and using two microcontrollers. CubeSec has redundant microcontrollers, for if one microcontroller fails the other microcontroller can take over the operation. Space, weight and power can be traded to get higher speed of encryption. Also a single microcontroller can be clocked at higher speeds using more power, without sacrificing space or weight, to get higher speed of encryption. Depending on the requirements, space, weight and power can be traded off at the cost of another.

Acknowledgements

This work was supported by the NSF Division of Electrical, Communications and Cyber Systems (ECCS), Integrative, Hybrid and Complex Systems (IHCS) Program, Grant #0901706.

Conclusion and future work

CubeSec is a low cost, configurable, small form factor (5cm x 5cm), low power (10mW) security solution for CubeSat communication. CubeSec uses 128 bit Advanced Encryption Standard (AES) and Data Encryption Standard (3DES) block encryption algorithms with Galois/Counter Mode (GCM). GndSec is the terrestrial counterpart of CubeSec. CubeSec and GndSec provides mutual authentication, confidentiality, data integrity between Cubesat and ground Station using pre-shared keys.

Future work will include Kerberos based security protocol suite for CubeSat to CubeSat and CubeSat to Ground Station authentication and secure data exchange in CubeSat Networks.

References

1. Cal Poly University, “Massive Operations, Recording, and Experimentation Database System (MoreDB),” http://moredbs.atl.calpoly.edu/
2. European Space Agency (ESA), Global Educational Network for Satellite Operations (GENSO),” http://www.genso.org/
3. Electronic Frontier Foundation, “Cracking DES,” O’Reilly & Associates, 1998.
4. S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, and M. Schimmler, “Breaking Ciphers with COPACOBANA - A Cost-Optimized Parallel Code Breaker,” Conference on Special-purpose Hardware for Attacking Cryptographic Systems, 2006.
5. Bo Yang, Sambit Mishra, Ramesh Karri, “High Speed Architecture for Galois/Counter Mode of Operation (GCM),” Information

Security: 10th International Conference, ISC 2007

6. Sren Rinne, Thomas Eisenbarth, Christof Paar, "Performance Analysis of Contemporary Light-Weight Block Ciphers on 8-bit Microcontrollers," 3rd International Symposium on Industrial Embedded Systems SIES 2008, pp. 58–66, 2008
7. Atmel, "AVR XMEGA Family of Microcontrollers, " "http://www.atmel.com/products/microcontrollers/avr/avr_xmega.aspx
8. Atmel, "Atmel ATXMega1281a microcontroller," http://www.atmel.com/devices/atxmega128a1.aspx
9. Atmel, "Atmel AVR Xplain," http://www.atmel.com/tools/AVRXPLAIN.aspx
10. Canonical Ltd., "Ubuntu Operating System", www.ubuntu.com
11. Free Software Foundation, Inc., "The GNU Compiler Collection (GCC)," http://gcc.gnu.org/
12. Eric Weddington, Brian Dean and others, "Avrdude: AVR Downloader/UploaDEr," http://savannah.nongnu.org/projects/avrdude/
13. Obulapathi, "CubeSec: a lightweight security solution for CubeSat communications," GitHub https://github.com/obulpathi/CubeSec