International Journal of Electronics and Communication Engineering.
ISSN 0974-2166 Volume 6, Number 2 (2013), pp. 187-197

http://www.irphouse.com

A Hybrid Fault Detection and Correction AES for Space Application

Kalaiarasi P.[1], N.D. bobby[2], C. Jayashree[3] and R. Lavanya[4]

[1,2,3,4]Assistant Professor, Department of ECE,
Vel Tech High Tech Engineering College
kalaiarasivlsi.12@gmail.com, ndbobby@gmail.com,
jaisrime80@gmail.com, lavanya_ecnics@yahoo.co.in

ABSTRACT

The demand to protect the sensitive and valuable data transmitted from satellites to ground has increased and hence the need to use encryption on board. The Advanced Encryption Standard, which is a very popular choice in terrestrial communications, is slowly emerging as the preferred option in the aerospace industry including satellites. Satellites operate in harsh radiation environment and therefore any electronic systems used onboard satellites such as processors, memories etc. are very susceptible to faults induced by radiation. So the encryption processor should be robust enough to faults in order to avoid corruption of valuable data and subsequent transmission to ground. This paper presents a novel model to detect and correct Single Event Upsets in on-board implementations of the AES algorithm, which is based on Hamming error correcting code. An FPGA implementation of the proposed model is carried out.

Keywords - Encryption, Advanced Encryption Standard, Hamming Code, Fault Tolerant.

INTRODUCTION

Recent unauthorized intrusions into satellite data have raised the importance of using security services on board [1]. Encryption, by far the most widely adopted security service in terrestrial networks, is used to protect data from unauthorized users. Although there are many encryption products and algorithms, the use of these products and algorithms on-board satellites has been overlooked until recently. But now satellite manufacturers are realizing the importance of on-board encryption to protect valuable data, especially after cases, which have proved that intrusion into

satellite data is not an impossible task [2, 3]. At present, more and more EO satellites are equipped with on-board encryption to protect the data transmitted to the ground station. However due to confidentiality and security reasons the coverage of this topic in the open literature is very limited. Examples of EO satellites that include on-board data encryption are the following: Space Technology Research Vehicle (STRV) -1d, Korea Multipurpose Satellite-II (KOMPSAT-2), MeteoSat Second Generation (MSG) Spacecraft, MetOp-A Polar Orbiting Spacecraft, Canadian satellite RADSAT-2 [4, 5, 6, 7]. The STRV, MetOp and RADSAT employ the Data Encryption Standard (DES) whereas KOMSAT uses the International Data Encryption Algorithm (IDEA). It can be seen from these examples that the encryption algorithms used in present satellite missions are proprietary or outdated algorithms like DES, rather than algorithms based on the latest encryption standards. The Rijndael algorithm approved as the Advanced Encryption Standard (AES) by the US National Institute of Standards and Technology (NIST) is a block cipher, which encrypts one block of data at a time.

To encrypt multiple blocks, modes of operation have been defined by NIST. AES is being adopted by many organizations across the world. Because of its simplicity, flexibility, easiness of implementation, and high throughput AES is used in many different applications ranging from smart cards to big servers, however at the time of the writing of this paper no use of AES on board satellites has been reported. In fact, hardware implementations of AES are well suited to resource-constrained embedded applications like satellites [8]. There are various hardware implementations of the AES algorithm on platforms like application specific integrated circuits (ASICs) and field programmable gate arrays (FPGAs) that achieve a significant throughput ranging from a few Mbit/s to Gbit/s [2, 9]. Thus the requirements of small EO satellites for high-rate data transmission are met by existing AES implementations. However, in addition to high throughput, immunity of the encryption process against faults is very important in satellites. Satellites operate in a harsh radiation environment and consequently any electronic system used on board, including the encryption processor, is susceptible to radiation-induced faults. Most of the faults that occur in satellite on-board electronic devices are radiation-induced bit flips called single event upsets (SEUs) [10, 11]. If faulty data is transmitted to the ground station, the user's request for data retransmission has to wait until the next satellite revisit period, with revisit time varying from a couple of hours to weeks. In order to prevent faulty during data transmissions, there is a need for an error detection and correction scheme during transmission. Satellite data can further get corrupted during transmission to ground due to noise in the transmission channel. The impact of radiation on semiconductor devices on board depends on orbit altitude, orientation, and time. In low Earth orbit (LEO) SEU rates of the order of 10¡6 bit¡1 day¡1 can be expected [12]. Static random access memory (SRAM) based FPGAs are particularly susceptible to SEUs [16].

For example, the estimated SEU rate of the XILINX Virtex FPGA device XQVR1000 placed in a satellite at an altitude of 1000 km and 60± inclination ranges from 0.4/h to 12.6/h depending on the solar flare activities [14]. Reliability is the most important issue in avionics design. SEUs must be detected and corrected on board before sending the data to ground. The triple modular redundancy (TMR) technique is one of the most widely used redundancy-based SEU mitigation techniques in

satellites. A TMR design consists of three identical modules, which are connected by a majority voting circuit to determine the output [11]. However, with the TMR technique the area and power overheads triplicate in comparison with the original module. The paper is organized as follows. Section II gives details about the AES algorithm. In Section III error detection and correction method is given. Section IV software simulation of AES algorithm and hamming codes is given. Section V concludes the paper.

The general over view of the proposed system in this paper is given bellow as a flow chart:

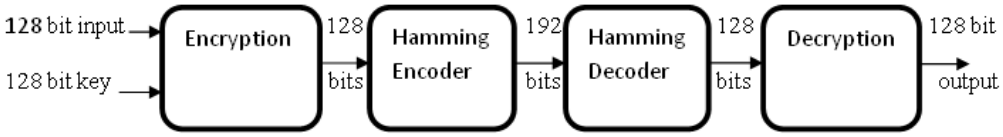

Figure 1. block diagram of proposed system

AES ALGORITHM

The AES algorithm is a symmetric block cipher that can encrypt and decrypt information. Encryption converts data to an unintelligible form called cipher-text. Decryption of the cipher-text converts the data back into its original form, which is called plain-text.

AES encryption

The AES algorithm operates on a 128-bit block of data and executed Nr - 1 loop times. A loop is called a round and the number of iterations of a loop, Nr, can be 10**,** 12, or 14 depending on the key length. The key length is 128, 192 or 256 bits in length respectively. The first and last rounds differ from other rounds in that there is an additional Add Round Key transformation at the beginning of the first round and no Mix Coulmns transformation is performed in the last round. In this paper, we use the key length of 128 bits (AES-128) as a model for general explanation. An outline of AES encryption is given in Fig. 1.

SubBytes phase

Each byte of the state is substituted with a 8-bit value from the S-box. The S-box contains a permutation of all possible 256 8-bit values. It is a nonlinear operation and the only non-linear transformation in this procedure. The S-box is gained by a multiplicative inverse over GF(28) and an affine transform .The sub bytes operation is required for both encryption and key expansion and its inverse is done for decryption. Its implementation has a direct impact on the overall throughput.

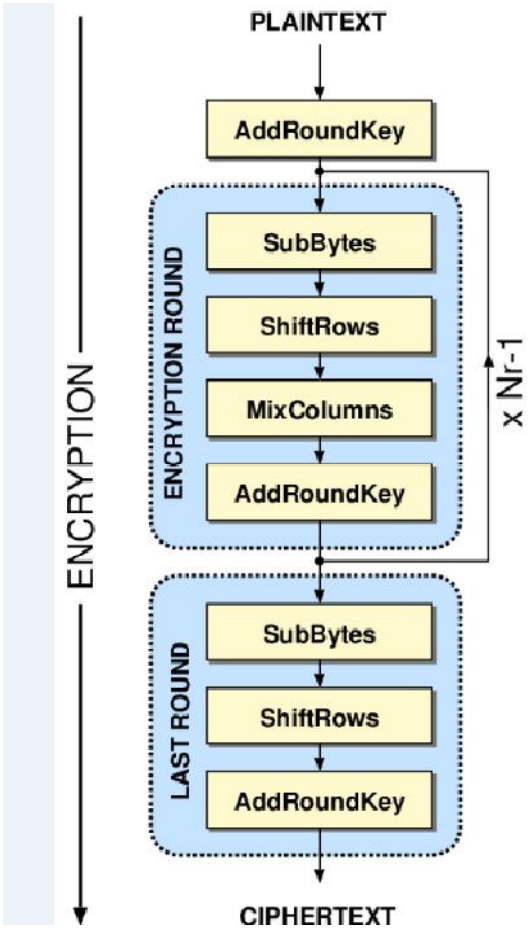

Figure 2.The AES-128 Encryption Algorithm

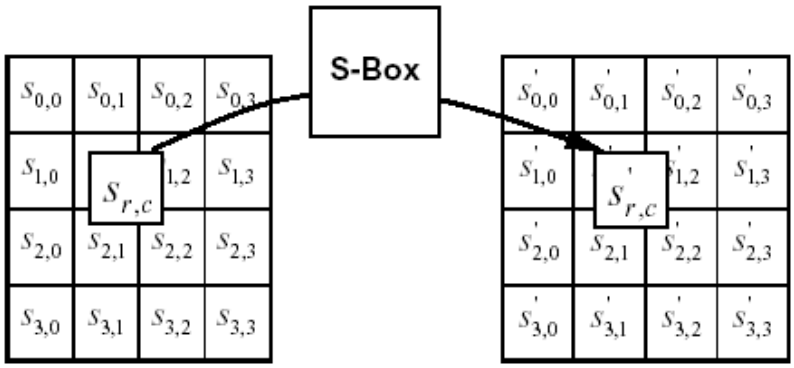

Figure 3.subbytes applies the sbox to each byte of the state

Shift Rows phase

In shift row operation, each row of the state is shifted cyclically to the left. The number of shift depends on the number of the row. The top row is not shifted and the last three rows are cyclically shifted over 1, 2, and 3 bytes, respectively.

MixColumn phase

MixColumn operation performs on the state column by column, and each column is treated as a four-term Polynomial over GF(28) .

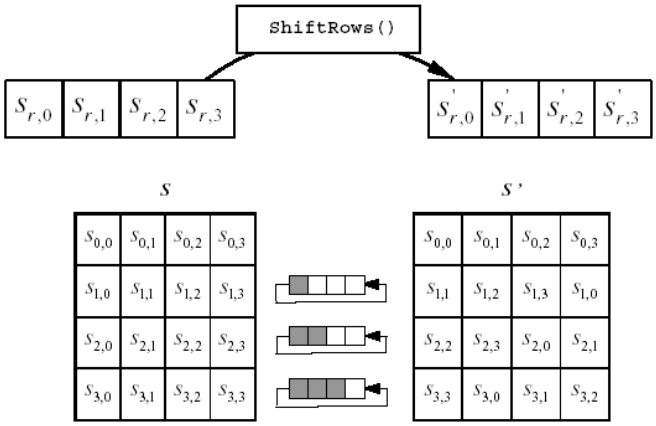

Figure 4.Shift rows cyclically shifts the last three rows in the state

As a result of this multiplication, the new four bytes in a column are generated as follows

A'=({02}.A)^({03}.B)^({01}.C)^({01}.D)
B'=({01}.A)^({02}.B)^({03}.C)^({01}.D) ----------> (1)
C'=({01}.A)^({01}.B)^({02}.C)^({03}.D)
D'=({03}.A)^({01}.B)^({01}.C)^({02}.D)

The operation of '^' is XOR operation modulo 2 and the '.' is a multiplication of polynomials modulo an irreducible polynomial m(x)=x8+x4+x3+x+1.The operation of {02}. *X* can be computed using Verilog HDL language:

{02}. *X* = {*X* [6: 0],1' b0}^(8 ' h1B&{8{ *X*[7]}})-------> (2)

So {03}_ *X* can be generated as follows:

{03}. X *=({02}.* X *)+{01}.* X *(3)*

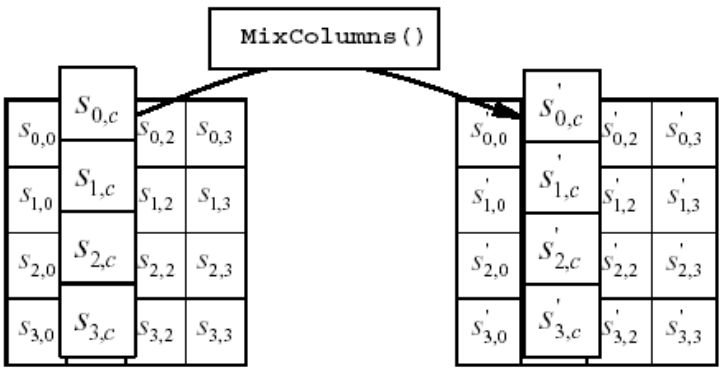

Figure 5. Mix columns operates on the state column-by-column

AddRoundKey phase

AddRoundKey operation is only a simple logical XOR of the state using a round key which is produced by the key expansion operation.

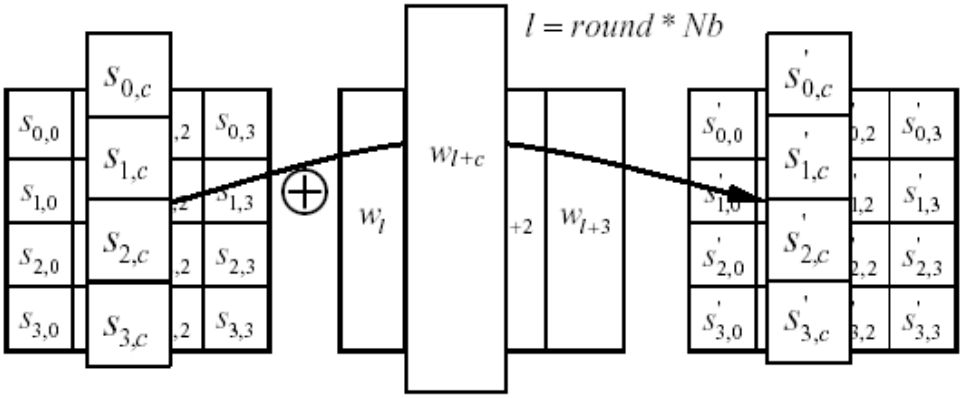

Figure 6. Add Round Key XORs each column of the state with a word from the key schedule

Key expansion phase

The key expansion operation generates a key schedule of 11 round-key of 16 bytes. Each of four consecutive bytes form a word, denoted wi .Taking this into account that the first round-key is the initial key and to generate every wi (except w0 -w3) the routine uses the previous w i-1 XOR w i-4 (except i mod 4 = 0).To get the wi, when the I mod 4 = 0,the operation has four stages, RotWord, SubWord, XOR Rcon[i / 4] and XOR w i-4. For the function RotWord a word [a0,a1,a2,a3] is the input, then performs a cyclic permutation, and returns the word [a1,a2,a3, a0] SubWord is a function that takes a four byte input word and applies the S-box to each of the four bytes to produce an output word. Rcon [i / 4],contains the values given by[xi / 4-1 ,{00},{00},{00}], with xi / 4-1 being powers of x is denoted as {02}) in the field GF(28)[5].Every following word, w[i],is obtained by performing XOR of the previous word, w[i-1],and the word Nk (Number of 32-bit words comprising the Cipher Key) positions earlier, w[i-Nk].

AES decryption

Decryption is a reverse of encryption which inverse round transformations to computes out the original plaintext of an encrypted cipher-text in reverse order. The round transformation of decryption uses the functions AddRoundKey, InvMixColumns, InvShiftRows, and InvSubBytes successively.

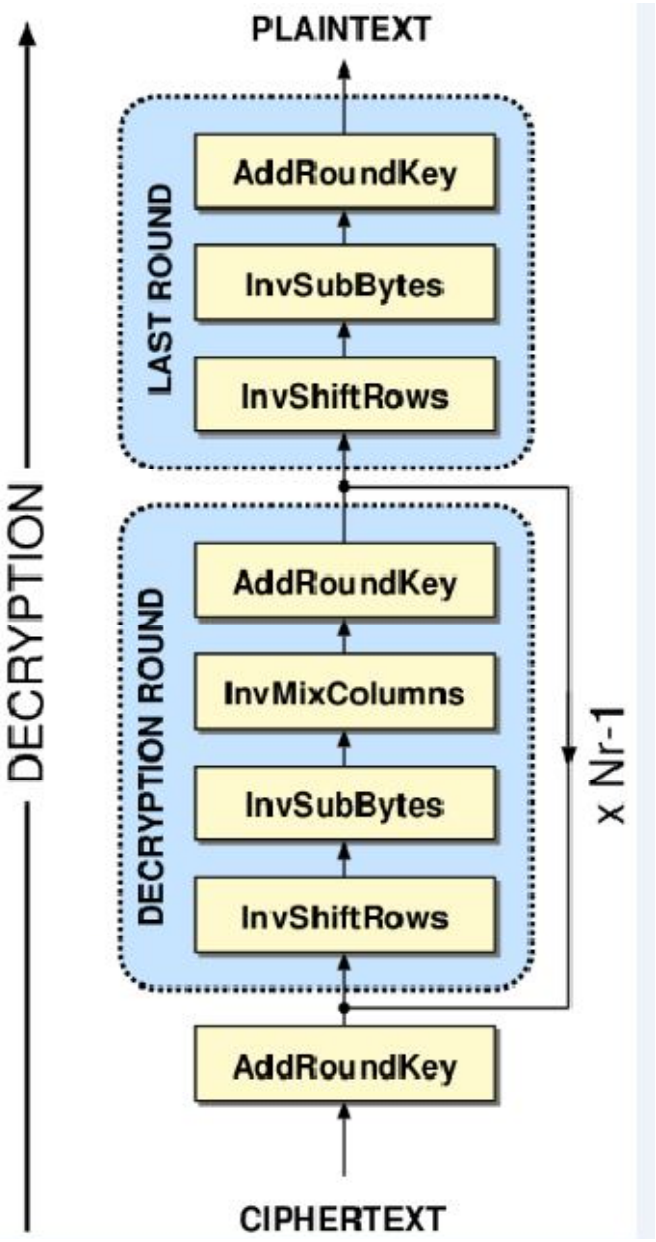

Figure 7.the AES decryption algorithm.

***AddRoundKey*:**
AddRoundKey is its own inverse function because the XOR function is its own inverse. The round keys have to be selected in reverse order. The description of the other transformations will be given as follows.

InvShiftRows Transformation:
InvShiftRows exactly functions the same as ShiftRows, only in the opposite direction. The first row is not shifted, while the second, third and fourth rows are shifted right by one, two and three bytes respectively.

***InvSubBytes transformation*:**
The InvSubBytes transformation is done using a once-precalculated substitution table called InvS-box. That InvS-box table contains 256 numbers (from 0 to 255) and their corresponding values.

InvMixColumns Transformation:
In the InvMixColumns transformation, the polynomials of degree less than 4 over GF(28), which coefficients are the elements in the columns of the state, are multiplied modulo ($x4 + 1$) by a fixed polynomial $d(x) = \{0B\}x3 + \{0D\}x2 + \{09\}x + \{0E\}$, where $\{0B\}$, $\{0D\}$; $\{09\}$, $\{0E\}$ denote hexadecimal values.

HAMMING CODES
When data is transmitted from one location to another there is always the possibility that an error may occur. There are a number of reliable codes that can be used to encode data so that the error can be detected and corrected. With this project you will explore a simple error detection-correction technique called a Hamming Code. A Hamming Code can be use to detect and correct one-bit change in an encoded code word. This approach can be useful as a change in a single bit is more probable than a change in two bits or more bits. Here is an example of how this process works. Consider the table below which has 12 positions. Data is represented (stored) in every position except 1, 2, 4 and 8. These positions (which are powers of 2) are used to store parity (error correction) bits.

1	2	3	4	5	6	7	8	9	10	11	12
		1		2	3	4		5	6	7	8

Figure 8. table with 4 parity position and 8 input positions.

Using the four parity (error correction bits) positions we can represent 12 values (1- 12). Using the format given, data is represented by the 8 non-parity bits. After placing the data in the table we find that in positions 3, 6, 9, 10 and 12 we have a '1' we obtain the binary representation for each of these values. We then exclusive OR the resulting values (essentially setting the parity bit to 1 if an odd # of 1's else setting it to 0). The parity bits are then put in the proper locations. This is the encoded code word that would be sent. The receiving side would re-compute the parity bits and compare them to the ones received. If they were the same no error occurred – if they were different the location of the flipped bit is determined. The parity bits re-calculation is done at the receiving end. The re-calculated parity information is then compared to the parity information sent / received. If they are both the same the result (again using an XOR – even parity) will be all 0's. If a single bit was flipped the resulting number will the position of the errant bit .

SOFTWARE SIMULATION

The design has been coded by Verilog HDL. All the results simulated in ModelSim – Altera 6.6d (Quartus **II** 11.0) software.

The results of simulating the encryption, hamming encoding, hamming decoding and decryption algorithm from the ModelSim simulator are shown in Fig.9, Fig.10, Fig.11 and Fig.12.

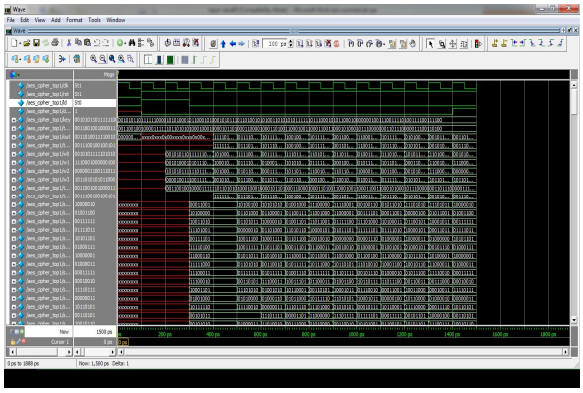

Fig.9. Output of encryption

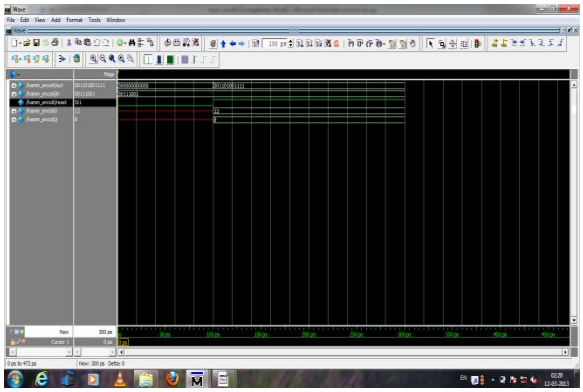

Fig.10. Output of hamming encoding.

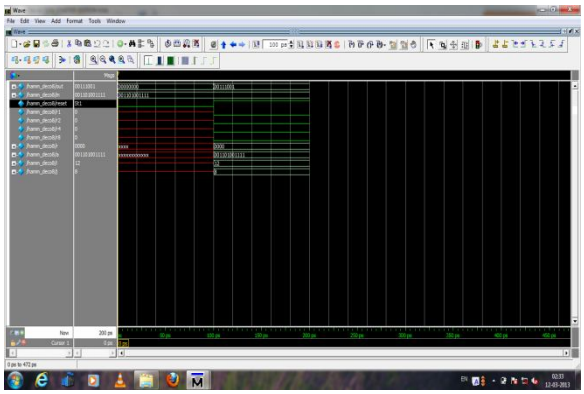

Fig.11. Output of hamming decoding.

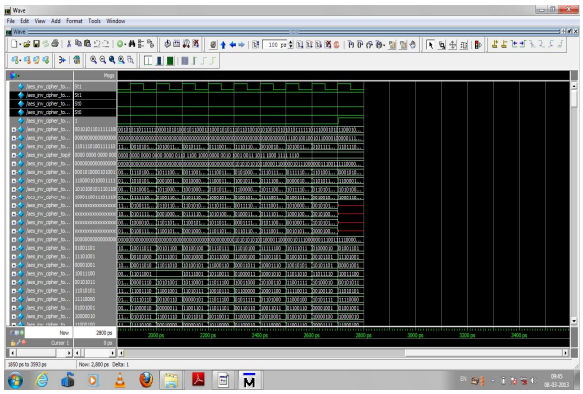

Fig.12. Output of decryption.

CONCLUSION

In the first phase of our paper, AES algorithm is used and encryption & decryption are performed. 128 bit cipher key used for encryption &decryption and ten rounds are performed to improve the security. In the second phase we are going to detect the errors occurring during transmission of the data and correct it .The third phase is implementing the fault detection and correction code and the original plain text is derived. All the above steps are simulated with ModelSim software and the output is got.

REFERENCES

[1] Consultative Committee for Space Data Systems Security threats against space missions. Informational Report CCSDS 350.1-G-1, Green book, NASA, Washington, D.C., Oct. 2006.

[2] Vladimirova, T., Banu, R., and Sweeting, M. N. On-board encryption in satellites. In Proceedings of the 8th Military and Aerospace Applications of Programmable Logic Devices and Technologies International Conference (MAPLD'2005), F-184, NASA, Washington, D.C., Sept. 2005.

[3] Sweet, K. The increasing threat to satellite communications. Online Journal of Space Communication, 6 (Nov. 2003)

[4] Weiss, H., and Stanier, J. Space mission communications security. In 5th Ground System Architecture Workshop (GSAW) 2001, http://sunset.usc.edu/events/GSAW/gsaw2001/SESSION9/Shave.pdf (last accessed 18th June 2007).

[5] Guttlich, J., Sinander, N., and Schaffner, E.MeteoSat second generation (MSG) ground segment–LRIT/HRIT mission specific implementation. Document EUM/MSG/SPE/057, 4.0, Sept. 21, 1999.

[6] Michalik, H., Hinsenkamp, L., and Schonenberg, A. Secure space links–Impacts on on-board link data processing. Data Systems In Aerospace (DASIA 2006), Berlin,Germany, May 22—25, 2006.

[7] Consultative Committee for Space Data Systems The application of CCSDS protocols to secure systems. Informational Report CCSDS 350.0-G-2, Green Book, NASA, Washington, D.C., Jan. 2006.
[8] Daemen, J., and Rijmen, R. The Design of Rijndael: AES–The Advanced Encryption Standard. New Yrok: Spriger-Verlag, 2002.
[9] Zhang, X., and Parhi, K. K.High-speed VLSI architecture for the AES algorithm. IEEE Transaction on VLSI Systems, 12, 9 (Sept. 2004), 957—967.
[10] Gussenhoven, M. S., and Mullen, E. G.Space radiation effects program: An overview. IEEE Transactions on Nuclear Science, 40, 2 (Apr. 1993), 221—227.
[11] Kastensmidt, F. L., Carro, L., and Reis, R. Fault-Tolerance Techniques for SRAM-Based FPGAs. New York: Springer, 2006.
[12] Underwood, C. I. The single-event effect behaviour of commercial-off-the-shelf memory devices–A decade in low Earth orbit. IEEE Transactions on Nuclear Science, 45, 3, Pt. 3 (June 1998), 1450—1457.
[13] Leon, A. F. Field programmable gate arrays in space. IEEE Instrumentation and Measurements Magazine, 6, 4 (Dec. 2003), 42—48.
[14] Fuller, E., Caffrey, M., Salazar, A., Carmichael, C., and Fabula, J. Radiation testing update, SEU mitigation, and availability analysis of the virtex FPGA for space reconfigurable computing. In Proceedings of Military and Aerospace Applications of Programmable Logic Devices and Technologies International Conference (MAPLD 2000), Sept. 26—28, 2000.