On-Board Security Services in Small Satellites

Tanya Vladimirova, Roohi Banu and Martin N. Sweeting

Surrey Space Centre
School of Electronics and Physical Sciences
University of Surrey, Guildford, UK, GU2 7XH
t.vladimirova@surrey.ac.uk; r.banu@surrey.ac.uk

Abstract

Security techniques for commercial satellites are insufficiently developed despite the explosive growth of security services in earthbound applications at present. Recently, the issue of satellite security has gained in importance in connection with attempts to intercept satellite data. This paper first gives a brief overview of security services, then security services required on board satellites, in particular Earth observation satellites, are identified and an on-board security architecture is proposed. The AES algorithm and the T-Box approach to AES are outlined. Hardware implementations of AES are reviewed and a novel fault-tolerant model of the AES algorithm is introduced. Software simulation results of the proposed fault-tolerant model are detailed.

Index Terms – Small Satellites, Earth Observation, Security Services, Encryption

1. Introduction

The history of cryptography stretches from the times of ancient Egypt to today and its importance is increasing day by day. In recent years, with the explosive advancement of computers and the Internet, the dependence of both organizations and individuals on the security of the information stored and communicated using these systems has increased. Security in military satellites is mandatory, and classified security products are used to protect the transmitted information. Security in commercial satellites has, however, been overlooked for various reasons such as limited computational resources, and partly due to the impression that satellites are very far and out of reach to hackers. Presently, satellite manufacturers are realizing the importance of security in satellites and the demand for security services in satellites is increasing steadily.

Recent intentional hack attacks have proved that intrusion into satellite data is not an impossible task. A team at the Embry Riddle Aeronautical University managed to obtain National Oceanic and Atmospheric Administration (NOAA) satellite imagery with basic apparatus built as part of an experimental project and by using open sources available from the Internet [1]. Similarly, researchers from a Japanese University were able to access data from NASA's Earth observation satellite LandSat as it flew over Japan [2,3]. Furthermore, the idea behind NASA's concept of a Space Internet [4] is that satellite users and scientists will directly access the satellite just like any other computer over the Internet to get the required information. Allowing direct access to spacecraft certainly gives flexibility, but at the cost of threats such as

unauthorized access and illegal use of valuable data. In order to prevent such problems, adequate security services are absolutely necessary.

Satellites are broadly classified into large and small satellites according to their mass. Satellites weighing less than 500 kg are classified as small satellites. The demand for small Earth observation (EO) satellites is growing. A network of small satellites in low Earth orbit (LEO) can provide an effective low-cost platform for remote sensing of various phenomena on Earth. The Disaster Monitoring Constellation (DMC) program is a novel international partnership comprising a network of five low cost micro-satellites and ground stations. The satellites are designed and manufactured by Surrey Satellite Technology Ltd (SSTL), UK, as a Know-How transfer to the participating countries: the United Kingdom, Nigeria, Algeria, Turkey and China. From a low Earth orbit, each satellite provides 32 metre multispectral imaging (green, red, infrared) over a 600 km swath width. The DMC program offers the possibility for daily revisiting of any point on the globe [5, 6].

Observation satellites demand increased data transmission rates. This requirement can be easily met by current hardware implementations of the Advanced Encryption Standard (AES) [7] which have achieved a throughput ranging from few Mbps to Gbps. However, in addition to high throughput, fault tolerance and reliability are undoubtedly key issues when designing encryption processors for space applications using either Field Programmable Gate Arrays (FPGAs) or Application Specific Integrated Circuits (ASIC) technologies.

In this paper we address security needs of small Earth observation satellites. The work is aimed at application on board a DMC satellite. The paper is organized as follows. Section 2 gives a brief introduction to the security services and summarizes the use of encryption in existing satellite missions. Section 3 discusses an on-board security architecture for EO small satellites. Section 4 describes briefly the AES algorithm and reviews AES hardware implementations. Section 5 introduces a novel fault-tolerant model of the AES algorithm. Section 6 gives details of the software simulation of the proposed fault-tolerant AES model.

2. Introduction to Security Services

A number of security services, such as confidentiality, integrity, authentication, non-repudiation, access control etc. are included in today's terrestrial security architectures to provide various security measurements at different abstraction levels.

- *Confidentiality* is used to keep the contents of information accessible to only those authorized to have it.
- *Integrity* is to make sure that data is not modified, deleted or inserted with some other data by unauthorized users.
- *Authentication* is concerned with assuring that the origin of a message is correctly identified.
- *Non-repudiation* prevents both the sender and the receiver of a transmission from denying previous commitments or actions.
- *Access control* is the security service that restricts access to information to authorized people only.

The security services, outlined above, are implemented using different security mechanisms such as encryption algorithms, hash functions, digital signatures etc. *Encryption* is the conversion of data into a form called ciphertext that cannot be easily understood by unauthorized users. Decryption is the process of converting encrypted data back into its original form so it can be understood. Encryption prevents any non-authorized party from accessing the data. In general there are two major classes of algorithms: symmetric (or secret) key and asymmetric (or public) key algorithms. Symmetric key encryption algorithms use the same key to encrypt and decrypt the data; whereas, asymmetric or public key algorithms use different keys. There are a wide variety of algorithms for symmetric and asymmetric key encryption. Table 1 summarizes popular asymmetric and symmetric key encryption algorithms [8, 9].

Table 1. Characteristics of Popular Encryption Algorithms

	Encryption Algorithm	**Mathematical Operations**	**Key Length (bits)**
SYMMETRIC	Data Encryption Standard (DES)	1. Initial Permutation 2. Expansion Permutation 3. S-box Substitution 4. Final Permutation 5. Key Generation	56
	Advanced Encryption Standard (AES)	1. AddRoundKey 2. ByteSub 3. ShiftRow 4. MixColumn	128 (supports also 192 and 256)
ASYMMETRIC	Rivest, Shamir Adleman, (RSA)	Modular arithmetic – Exponentiation, Multiplication.	1024 – 15,360
	Elliptic Curve Cryptography (ECC)	Galios Field modular arithmetic	163 – 571

In general symmetric key encryption algorithms are used for bulk data encryption, as public key encryption algorithms require a lot of computational resources. DES is easily breakable because of the short key length. AES is the latest encryption algorithm, which is suitable for a variety of platforms ranging from smart cards to big servers. AES is fast in both software and hardware, is relatively easy to implement, and requires little memory [10]. Asymmetric key encryption algorithms are based on mathematical functions rather than on substitution and permutation. The use of two separate keys has profound consequences in the areas of key distribution and authentication [8]. In terrestrial applications, the RSA algorithm is a very popular choice especially for digital signatures, however, it features large key sizes. For small

satellites the ECC is a more suitable choice [11, 19] as it has a smaller key size and hence is appropriate for resource-constrained applications.

Hash function is a transformation that maps a message of any length into a fixed length, which is called hash value or message digest used for integrity. *Digital Signature* is an electronic rather than a written signature that can be used by someone to authenticate the identity of the sender of a message or of the signer of a document. It can also be used to ensure that the original content of the message or document that has been conveyed is unchanged. The relationship between security services, mechanisms and cryptography algorithms is illustrated in Table 2.

Table 2. Security Services, Mechanisms and Algorithms

Security Service	**Security Mechanism**	**Algorithms**
Confidentiality (Encryption)	Symmetric and Asymmetric Key Algorithms	DES, 3-DES, IDEA, AES, RSA, ECC
Authentication	MAC and Digital Signatures	MD5, SHA-1, KEBROS, DSA
Integrity	Hash Functions, Digital Signatures, Checksum Function	RSA, DSA, ECC

At present, only very few EO satellites are equipped with on-board security services, in particular only encryption services are used to protect the data transmitted to the ground station. Security services such as authentication and data integrity, which are required for the overall protection of satellite data, are not addressed at present. This has been highlighted in the United States General Accounting Office report (GAO-02-781) [2] on mitigating the risk of satellites being taken over by unauthorized users. Table 3 summarizes the use of encryption in current satellites. It can be seen from Table 3 that the encryption algorithms used in present satellite missions are proprietary or outdated algorithms like DES, rather than the latest encryption standards.

Table 3. Summary of the Use of Encryption in Current Satellite Missions

Spacecraft Name	**Algorithms Used**	**Implementation Platform**	**Encrypted Data**
Space Technology Research Vehicle (STRV) 1c/d [13]	Data Encryption Standard (DES)	Software (on SPARC processor)	Low rate downlink
MeteoSat Second Generation (MSG) Spacecraft [14]	EXOR Function	Hardware	High rate downlink
Korea Multipurpose Satellite II (KOMPSAT-II) [16]	International Data Encryption Algorithm (IDEA)	Hardware	High rate downlink
METOP Polar Orbiting Spacecraft [14]	Triple Data Encryption Standard (3-DES)	Hardware	High rate downlink

3. On-Board Security Architecture for Earth Observation Small Satellites

In general, an EO small satellite consists of a number of subsystems and imaging payloads. All the subsystems are interconnected through an on-board bus or network. The receiver block in the communications subsystem receives the commands from the ground station, demodulates and decodes the uplink signals and generates telecommands. The transmitter encodes and modulates the data collected from the on-board subsystems and transmits them to the ground station. Usually, the downlink consists of two parts: low rate transmitters are used to downlink telemetry and high rate transmitters are used to transmit the bulk imaging data stored in the mass memory of the payload. For the high rate downlink, data need to be encrypted on demand by the ground station during the contact period and therefore the encryption process should be high-speed to achieve real-time transmission.

In order to secure the communication between the satellite and the ground station, both the uplink and the downlink need to be protected. In addition, similar to secure terrestrial architectures, all security services like authentication, integrity and encryption should be used for complete protection of the satellite communication links. The uplink or telecommand channel should be checked for integrity and authentication in order to protect the satellite from being taken over by unauthorized people. Both the high rate and the low rate downlinks of Earth observation satellites should be encrypted to protect the valuable and sensitive data transmitted to the ground station. The low rate downlink consists of sensitive information such as satellite control information (attitude and orbit information), on-board voltage and temperature measurements and time stamps.

Fig. 1 shows a block diagram of our proposed on-board security architecture for a small EO satellite. For the sake of simplicity Fig. 1 includes only the main connections between the subsystems. The following security blocks are introduced to protect the communication links: *an authentication and integrity check block, an encryption block* and *a real-time high-speed encryption block*. The integrity and authentication block in Fig. 1 provides protection to the satellite by ensuring that the on-board data handling subsystem receives telecommands from an authorized ground station. Any telecommands that do not pass the authentication process are rejected and reported back to the ground station through the downlink. Security mechanisms like hash functions and digital signatures should be used to achieve authentication and integrity of telecommands as discussed in section 2. The purpose of the encryption block in Fig. 1 is to encrypt the low rate telemetry downlink, while the real-time high-speed encryption block in Fig. 1 protects the high rate downlink channel.

4. The Advanced Encryption Standard

The Rijndael algorithm, which was approved as the Advanced Encryption Standard by the US National Institute of Standards and Technology (NIST) in October 2000, is being adopted by many organizations across the world. AES is well suited to resource-constrained applications, such as small satellites because of its simplicity, flexibility, easiness of implementation and high throughput [7]. In fact, AES is gradually emerging as the preferred algorithm in the aerospace industry. The

Consultative Committee for Space Data Systems (CCSDS) is considering recommending AES as the standard encryption algorithm for use on satellites [16].

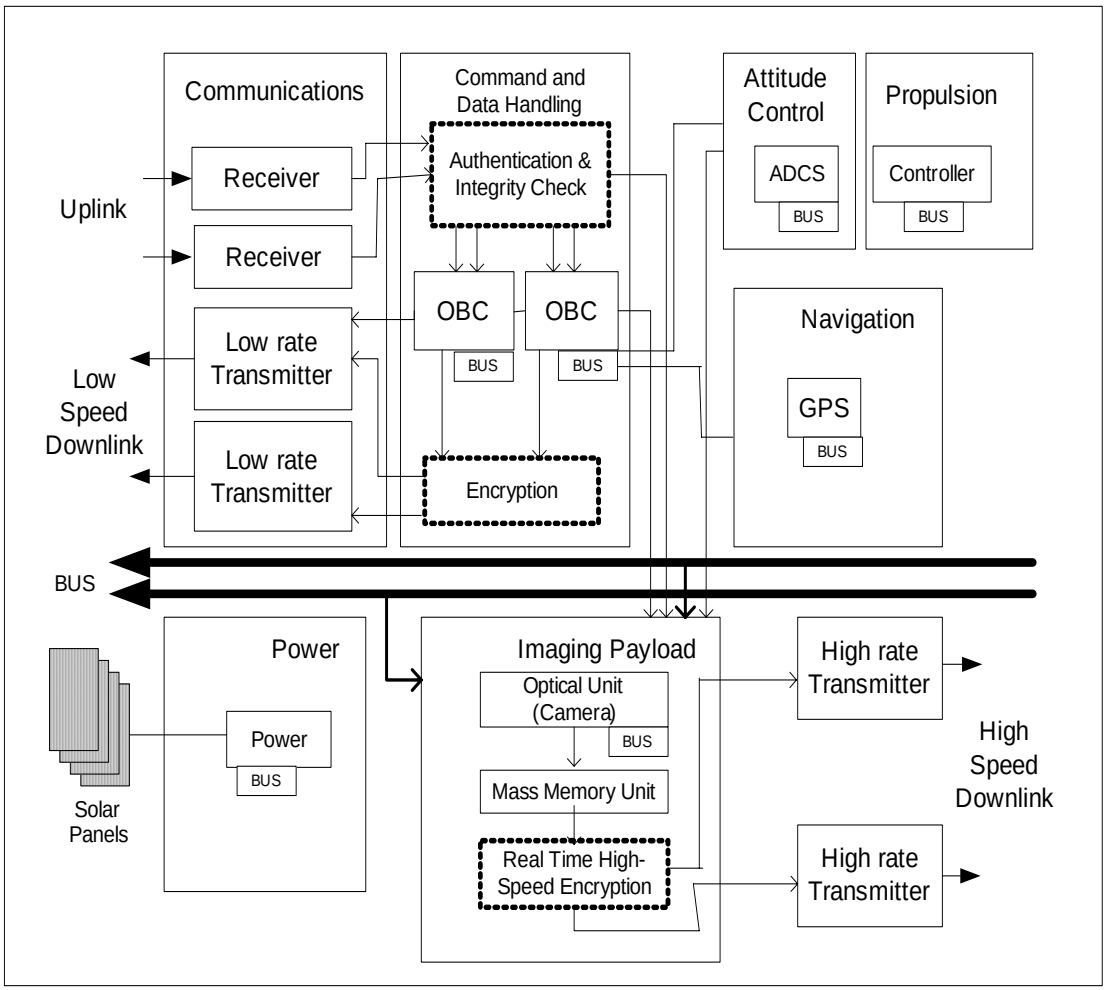

Figure 1. Block Diagram of the Proposed On-Board Security Architecture

4.1 The AES Algorithm

The AES is a symmetric key algorithm, in which both the sender and the receiver use a single key for encryption and decryption. The standard defines the block length to 128 bits and supports key lengths of 128, 192 or 256 bits [17]. The AES is an iterative algorithm and each iteration is called a round. The total number of rounds (N_r) is 10,12, or 14 when the key length is 128,192 or 256 bits respectively. Each round in AES except the final round consists of four transformations: *SubBytes*, *ShiftRows*, *MixColumns* and *AddRoundKey*, while the final round does not have the *MixColumns* transformation as shown in Fig. 2. A round transformation of AES and its steps operate on some intermediate results, called state. The state can be visualized as a rectangular matrix with four rows. The number of columns in the state is denoted by Nb and is equal to the block length in bits divided by 32. For a 128-bit data block the length Nb is 4.

The *SubBytes* transformation is a non-linear byte substitution, operating on each byte of the state matrix independently. This transformation can be calculated on the fly for each state byte. However, for reasons of efficiency, in most practical implementations of AES *SubBytes* are computed in advance and stored in a look-up table (LUT) of 2^8 (= 256) elements called an S-Box (S_{RD}) table. *ShiftRows* cyclically left shifts the last three rows of the state by 1, 2 and 3 bytes respectively. *MixColumns* transforms every column in the state by multiplying it with a predefined polynomial [2 3 1 1]. Finally, the *AddRoundKey* transformation adds the expanded round key to the state by an XOR operation [17]. The transformations involved in the key expansion use operations like substitution (*SubBytes*), shift and XOR, which are similar to those comprising the AES transformations and therefore they are not discussed in detail.

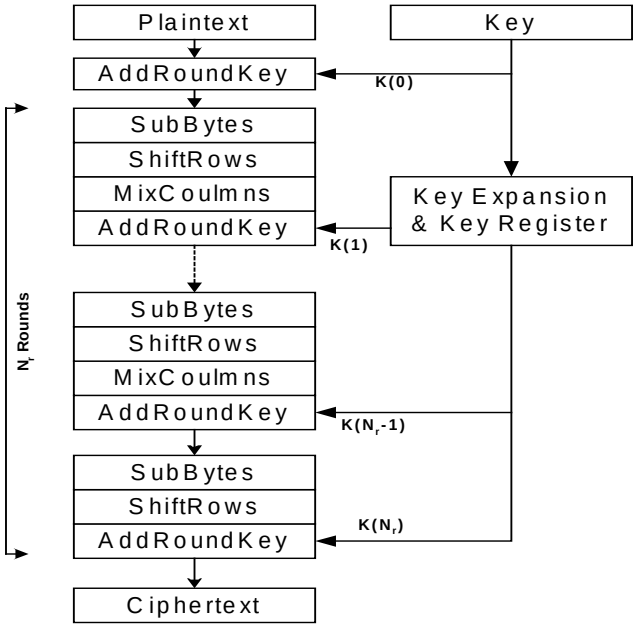

Figure 2. Block-Diagram of the AES Algorithm

The state of each round using the above four transformations can be expressed mathematically as follows:

$$\begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} = \begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} S_{RD}[a_{0,j+C_0}] \oplus \begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} S_{RD}[a_{1,j+C_1}] \oplus \begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} S_{RD}[a_{2,j+C_2}] \oplus \begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} S_{RD}[a_{3,j+C_3}] \oplus K_j$$

$$0 \le j < 4$$

.....(1)

Where *a* represents the input at the beginning of the round transformation S_{RD} is the S-Box look-up table and K_j is the round key in round j. C_0, C_1, C_2, C_3 are 0, 1, 2, 3, respectively, to account for the amount of shifting in the *ShiftRow* operations.

The AES data path can also be implemented using a T-Box approach [17]. In the T-Box approach *SubBytes* and *MixColumns* transformations are merged into a single transformation to achieve high speed and to simplify the implementation as shown in Fig. 3. The T-box approach implements the combination of *SubBytes* and *MixColumns* transformations by look-up tables. Instead of storing only the value of *SubBytes* in the S-box approach, the T-box approach stores the values of *SubBytes* (S_{RD}), $(S_{RD} \otimes \{02\})$ and $(S_{RD} \otimes \{03\})$, where $\otimes$ represents multiplication in Galois Field.

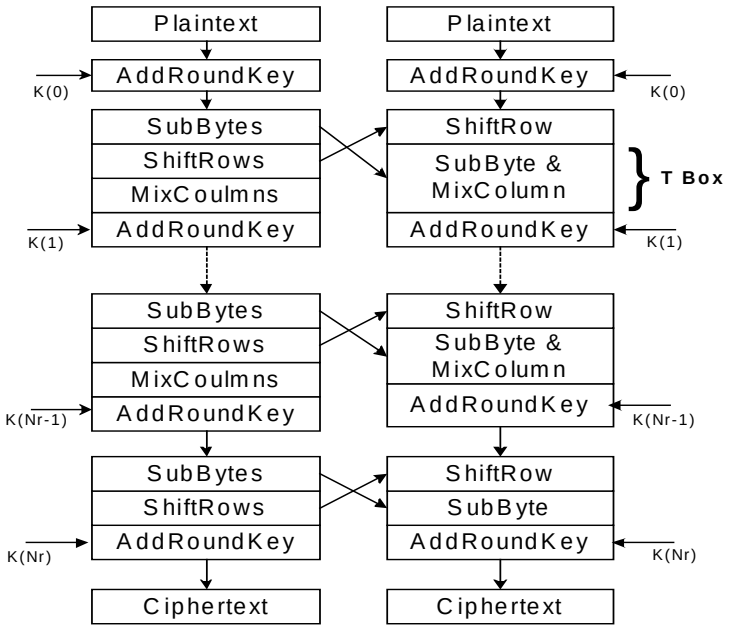

Figure 3. Block-Diagram of the AES Algorithm using the T-Box Approach

4.2 Hardware Implementations of AES

The AES algorithm has been a focus of intensive hardware design effort. Table 4 summarises recent FPGA and ASIC implementations of AES. It can be seen from Table 4 that AES is able to achieve a significant throughput ranging from Mbps to Gbps while occupying a relatively small chip area. Designers have sought different optimisation goals by applying various architectural approaches and trading off between LUT and non-LUT implementations of the *S-Box* and *MixColumns* transformations [18,19,20].

The AES algorithm has been selected for the development of the encryption blocks in Fig. 1, which are to be implemented in hardware. Evaluation of existing AES intellectual property (IP) cores is in progress to identify suitable candidates for FPGA prototyping. One such candidate core is the open-source AES IP core, provided at the OPENCORES website (www.opencores.org), which is written in the Verilog hardware description language. This IP core uses an LUT approach for the *SubBytes* transformation and a non-LUT approach for the *MixColumn* transformation. The core achieved a maximal throughput of 246 Mbps based on back-annotated simulation in ModelSim for the XILINX Virtex II device XC2V1000, which satisfies the performance requirements of our application.

Table 4. AES Hardware Implementation Survey

Author and Publication Year	Device or Technology	Device Utilization	Throughput Mbps	Algorithmic Optimisations	Architectural Optimisations	Implementation Target
McLoone 2003	XC800E	468 slices 20 BRAM	310	S-Box → LUT MixColumn → Non-LUT	Iterative architecture	To implement AES for 128,192,256 bits of data and key sizes
Jhing-Fa 2003	XCV812E	3,046 slices 280 BRAM	1,952	S-Box → LUT MixColumn → LUT	Pipelining	Complete implementation using look-up tables
Shuenn 2004	XCV1000E	1,857 slices 395 BRAM	1,604	S-Box → LUT MixColumn → Non-LUT	Iterative architecture	To implement both encryption/decryption in hardware
Henry 2002	CMOS 0.18µm	613 K gates	2,290	S-Box → LUT MixColumn → Non-LUT	Loop-unrolling	To achieve high throughput without pipelining
Yeong 2004	CMOS 0.25µm	80 K gates	1,454	S-Box → Non- LUT MixColumn → Non-LUT	Sub-pipelining	To achieve high throughput for memory-less devices using full pipelining
Rodriguez 2003	XCV2000E	5,677 slices 80 BRAMS	4,121	S-Box → LUT MixColumn → Non-LUT	Pipelining	To achieve high throughput
Gaj 2003	XC2S30-6	222 slices 3 BRAM	166	S-Box → LUT MixColumn → Non-LUT	Iterative architecture	The target is low gate count to achieve low-power
Gaj 2004	XC3S50-4	163 slices	208	S-Box → LUT MixColumn → Non-LUT	Iterative architecture	The target is low gate count to use cheap FPGAs
	XC2V40-6	146 slices	358			
Zhang 2004	XCV1000-6	11,014 slices	16,032	S-Box → Non-LUT MixColumn → Non-LUT	Sub-pipelining	To achieve very high throughput
	XCV1000e-8	11,022 slices	21,556			

5. Fault-Tolerant Model of AES

Satellites operate in the harsh radiation environment of space and therefore any electronic systems used on board, such as processors, memories as well as crypto-processors are susceptible to faults induced by radiation. In addition, a single bit fault occurring during the encryption process can propagate and cause multiple faults in the encrypted data. In fact, on average 50 % of the bits in the final encrypted data could be corrupted as a result of a single bit fault during encryption [21].

Faults must be detected and corrected on board, before sending the data to ground to avoid redundant transmission and use of erroneous data. Also, if faulty data is transmitted to the ground station, the user's request for data re-transmission has to wait until the next satellite revisit period, with revisit times varying from a couple of hours to weeks. Most of the faults that occur in satellite on-board electronic devices are radiation induced single bit flips called Single Event Upsets (SEU) [22]. SEUs are soft or temporary faults and correcting them can restore the normal operation of the device. Therefore special measures should be taken to protect on-board encryption processors from faults. In this section, a novel fault-tolerant model of the AES using the Hamming error correcting code is proposed to mitigate SEUs in on-board AES implementations.

The AES fault detection models described in the literature can be classified into two categories:

- Parity based [21,24] - the fault is detected by comparing the predicted parity with the calculated parity at the end of each transformation.
- Redundancy based [23] - a decryption module is used in parallel with the encryption module and its output is compared with the input to the encryption module to detect a fault.

For space applications fault detection alone is not enough and it is equally important to have fault correction capability on board. No fault correction schemes for AES were found in the literature at the time of writing of this paper. The rest of this section details our proposed fault-tolerant model of AES, which is based on the T-Box scheme, described in section 4.1. The model provides fault detection and correction functionality in the data path of AES using the parity error detection code and the Hamming error correcting code at byte level.

5.1 AES Fault Detection

The existing methods for fault detection of AES are mainly aimed at avoiding cryptanalysis of AES by injection of faults. Karpovsky et al. [25] developed a method to reduce the number of intentional undetected faults using systematic non-linear robust error detection codes. Bertoni et al. [21] proposed a fault-detection scheme based on the parity error detection code (EDC). For parity prediction in the *SubByte* transformation an additional bit is added to each element in the S-Box to represent the predicted parity. For the remaining transformations parity is predicted on the fly. Wu et al. [24] proposed a low cost concurrent error detection technique for the AES using parity checking based on Substitution Permutation Networks (SPN) with the aim of reducing the overhead for fault detection. The SPN approach is very efficient for fault detection, however, it will not help to locate the fault at byte level, which is essential for fault correction. Karri et al. [23] proposed a redundancy-based technique. The fault detection model exploits the inverse relationship between encryption and decryption to detect the fault at the algorithm level, round level and individual transformation level. The hardware overhead in this approach is very high, close to that of duplication either in space or time. Error detecting codes are a reasonable alternative to the duplication technique due to reduced hardware overhead, optimum detection rate and many degrees of freedom in choosing the desired error coverage/cost trade-off [26].

Our proposed fault detection approach uses a parity prediction method similar to [21,24], however, its implementation is based entirely on a look-up table approach. The advantages of the LUT approach are as follows:

- Simplicity of implementation
- The parity tables are treated as separate entities so that memory fault protection techniques can be achieved easily, which is essential in space applications.
- The model can be extended to enable fault correction as described in section 5.2 below.

As discussed in section 4, the AES round transformations operate on intermediate results, called states, which can be represented in the form of 4 x 4 arrays. We propose a fault detection scheme, which is illustrated in Fig. 4. The fault detection, which is carried out in each transformation of every AES round, consists of three steps. The first step involves prediction of the parity matrix using the input state to the transformation. The parity is predicted using parity LUTs corresponding to the transformations of the T-Box approach. The second step involves calculation of the parity matrix using the output from the transformation. The third step involves comparison of the predicted parity matrix with the calculated matrix to detect the fault.

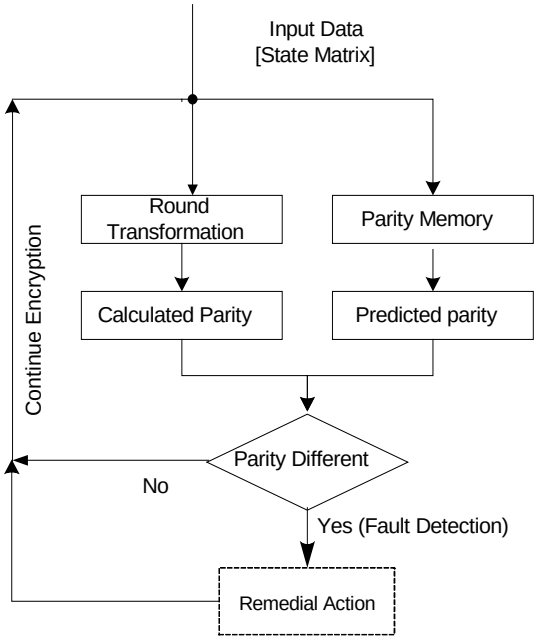

Figure 4. Flow Chart of Fault Detection in AES

5.2 AES Fault Detection and Correction Model Based on the Hamming Code

We have developed a new AES model, which incorporates in-path error detection and correction (EDAC) provisions, making it immune against SEUs. The scheme is based on the parity-based fault detection scheme, described above, with the addition of a fault correction capability. The current version of the model employs the Hamming (12,8) error correction code. The Hamming code (12,8) detects and corrects single bit faults in a byte, however, the correction model can be extended to provide for multiple bit faults by using sophisticated error correction codes such as Read-Solomon codes etc..

The procedure of fault detection and correction using the Hamming code includes four steps, as shown in Fig. 5. The first step involves predicting of the Hamming code matrix using the input state to the transformation. The Hamming code is predicted using the Hamming code resulting from the transformations of the T-Box approach. The second step involves calculation of the Hamming code matrix using the output from the transformation. The third step involves comparing of the Hamming codes to detect the fault. The final step involves correction using the predicted Hamming code to correct the faulty bit.

6. Simulation Results

The AES fault-tolerant model was verified with a purpose-built software simulator written in the JAVA programming language. The verification was implemented via injecting faults randomly at different levels: round, transformation, byte and bit level. The software simulator was able to detect all the faults up to bit level as expected using the Hamming codes (12,8). Both the fault detection and correction models were

subjected to extensive testing with test vectors provided by NIST [27], which proved the validity of the proposed AES error detection and correction scheme.

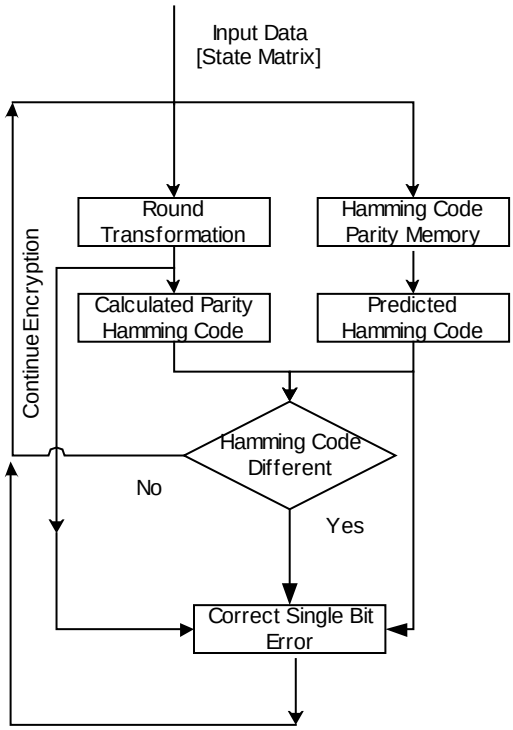

Figure 5. Flow Chart of Fault Correction in AES

A Graphical User Interface (GUI) was also developed to display and manage the procedures of fault injection, detection and correction (Fig. 6 and Fig. 7). The graphical user interface contains three sub-frames – *input*, *inject error* and *details*. The *input* sub-frame displays the input data block, encryption key, cipher block, decipher block etc.. The *inject error* sub-frame is used to simulate the error injection at different levels: round, transformation, byte and bit position. The *details* sub-frame shows the intermediate state of the output for every transformation and for every round in AES. The *details* sub-frame also shows the predicted and calculated parity or the Hamming code.

Fig. 6 shows a snapshot of the AES simulator window, which illustrates the fault detection process, based on the technique in section 5.1. The status bar at the lower end of the screen displays the result of the fault detection at byte level of a 1-bit error injected at round number 3, in the *ShiftRows* transformation, in the 5^{th} byte position of the state. Fig. 7 shows a snapshot of the AES simulator window, which illustrates the fault detection and correction process based on the technique introduced in section 5.2. The status bar at the lower end of the screen displays the result of the fault detection process at bit level of a 1-bit error injected at round number 3 in the *ShiftRows* transformation, in the 5^{th} byte position of the state, at the 6^{th} bit position.

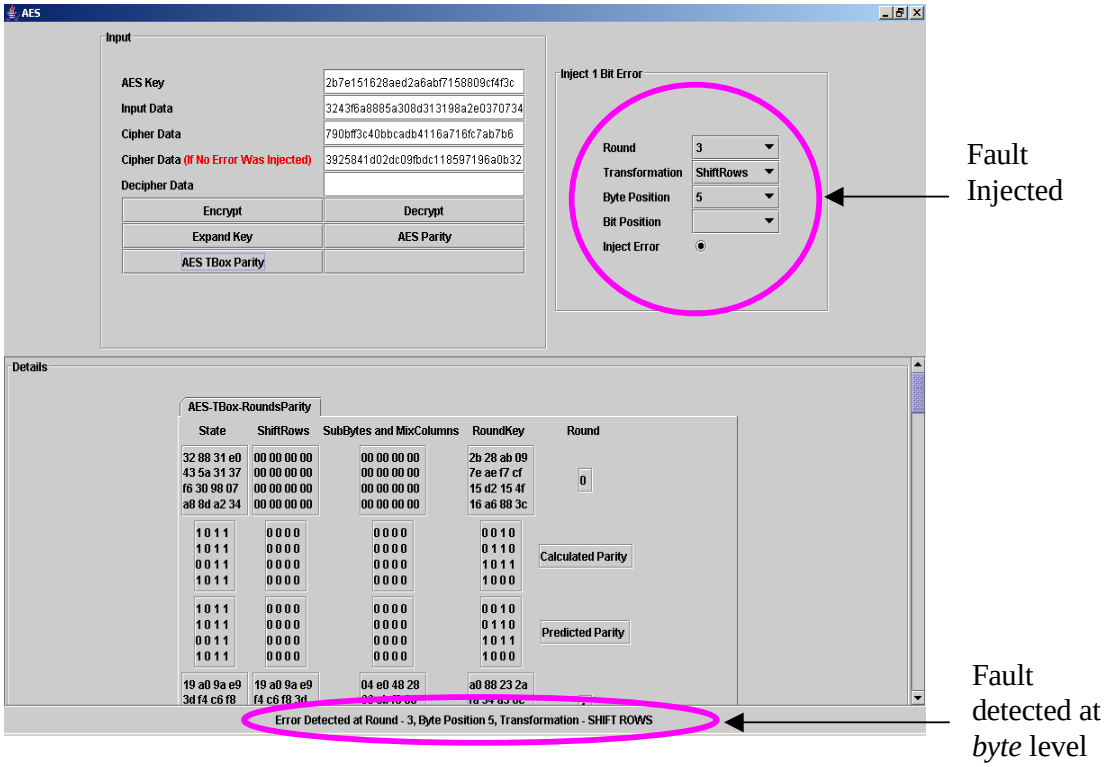

Figure 6. JAVA GUI for Fault Detection of AES

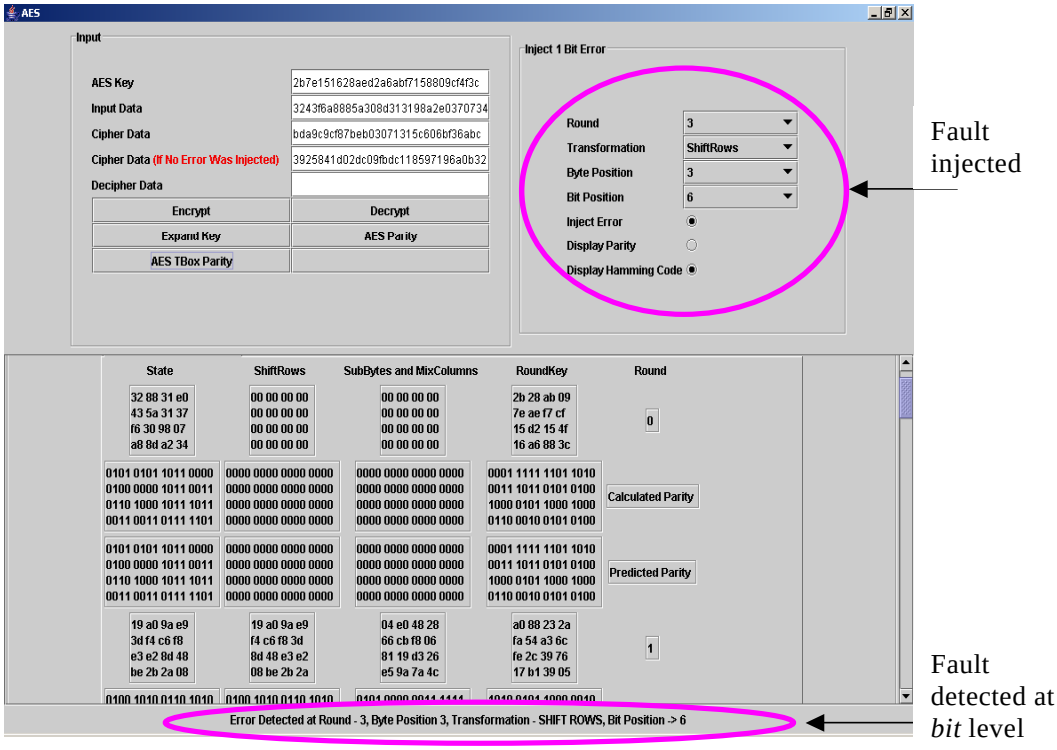

Figure 7. JAVA GUI for Fault Detection and Correction of AES

5. Conclusions

In this paper we address the need for on-board security services in satellites. An on-board security architecture for small EO satellites is proposed and the required security services are identified along with suitable algorithms for on-board use. The AES is identified as a suitable encryption algorithm for on-board use in small satellites. Aspects related to hardware implementation of the AES algorithm are discussed.

An AES fault detection model based on parity prediction is developed and verified by software simulation. A novel fault-tolerant model of AES using the Hamming error correcting code is proposed to mitigate SEUs or single bit faults in AES on-board implementations. The model is fully verified by means of purpose-built simulation software in Java. The proposed AES fault detection and correction model can also be used in other harsh radiation environments, for example in unmanned aerial vehicles, etc..

6. References

[1] K. Sweet, "The Increasing Threat to Satellite Communications" *Online Journal of Space Communication*, Issue 6, November 2003.
[2] "Critical Infrastructure Protection. Commercial Satellite Security Should Be More Fully Addressed", US Government Accountability Office Report, GAO-02-781, August 2002.
[3] K. Poulsen, " Satellites at Risk of Hacks", *Security Focus*, Oct 2002,
URL : http://www.securityfocus.com/news/942
[4] NASA/GSFC. "IP-in-Space Security Handbook", September 2001. URL:http://ipinspace.gsfc.nasa.gov/documents/
[5] W. Sun, P. Stephens, M. N. Sweeting. "Micro-Minisatellites for Affordable EO Constellations - RapidEye and DMC"- Proceedings of the IAA Symposium on Small Satellites for Earth Observation, Berlin, IAA-B3-0603, April 2001
[6] "Surrey Missions: DMC – Disaster Monitoring Constellation", Datasheet, SSTL,
URL: http://www.centaur.sstl.co.uk/datasheets/Mission_DMC.pdf
[7] B. Olsen. "Encryption Gets a Boost", *Federal Computer Week*, August 2004,
URL: http://www.fcw.com/fcw/articles/2004/0823/feat-encryptn-08-23-04.asp
[8] W. Stallings "Cryptography and Network Security – Principles and Practices", 3rd edition, Prentice-Hall, 2002.
[9] B. Schneier, "Applied Cryptography – Protocols, Algorithms and Source Code in C", Second Edition.
[10] Advanced Encryption Standard, Wikipedia,
URL:http://en.wikipedia.org/wiki/Advanced_Encryption_Standard
[11] K. Lauter, "The advantages of Elliptic Curve Cryptography for Wireless Security", Kristin Lauter, *IEEE Wireless Communications*, February 2004.
[12] T. Wollinger, J. Guajardo, C. Paar, " Cryptography in Embedded Systems: An Overview", *Proceedings of the Embedded World 2003 Exhibition and Conference*, pp. 735-744, Design and Electronik, Nurenberg, Germany, February 2003.
[13] H. Weiss, J. Stanier, "Space Mission Communications Security, GSAW 2001,
URL: http://sunset.usc.edu/events/GSAW/gsaw2001/SESSION9/Shave.pdf

[14] MeteoSat Second Generation (MSG) Ground Segment – LRIT/HRIT Mission Specific Implementation, Doc No. MSG/SPE/057, Issue 4.0, 21st September 1999.
[15] KOMPSAT II –
URL: http://www.ohb-system.de/gb/Satellites/OurContribution/kompsat2.html
[16] Consultative Committee for Space Data Systems – "The application of CCSDS protocols to secure systems". CCSDS 350.0.G-1. Green Book. March 1999.
[17] J. Daemen and R. Rijmen, "The Design of Rijndael: AES – The Advanced Encryption Standard", 1st edition, Spriger-Verlag, 2002.
[18] X. Zhang, K.K. Parhi . "Implementation Approaches for the Advanced Encryption Standard Algorithm", *IEEE Circuits and Systems Magazine*, Vol. 2, No 4, Fourth Quarter 2002, pp. 24 - 46
[19] X. Zhang, K.K. Parhi, "High-Speed VLSI Architecture for the AES Algorithm", *IEEE Transaction on VLSI*, Vol. 12, No 9, September 2004, pp. 957 – 967.
[20] F. Rodriguez, " 4.2 Gbits/s single-chip FPGA implementation of AES algorithm", Electronics Letters, Vol. 39, No.15, July 2003.
[21] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri and V. Piuri, " Error Analysis and Detection Procedures for a Hardware Implementation of the AES", *IEEE Transactions on Computers*, April 2003.
[22] C. I. Underwood, " An Investigation into the Suitability of COTS Technology for Aerospace Missions: Robust COTS-Based Architectures – Experience and Approaches from Space-Flight", SPACERAIN Report, Surrey Space Centre, 2005.
[23] R. Karri, K. Wu, P. Mishra, "Concurrent Error Detection Schemes for Fault-Based Side-Channel Cryptanalysis of Symmetric Block Ciphers", *IEEE Transactions on Circuits and Systems*, Dec 2002.
[24] K. Wu, R Karri, G. Kuznetsov, M. Goessel, "Low Cost Concurrent Error Detection for the Advanced Encryption Standard", *Proceedings of the International Test Conference*, ITC, 2004.
[25] M. Karposvsky, K. J. Kulikowski, A. Taubin," Robust Protection against Fault-Injection Attacks on Smart cards Implementing the Advanced Encryption Standard", *Proceedings of the 2004 International Conference on Dependable Systems and Networks*, *DSN 04*, 2004.
[26] L. Breveglieri, I. Koren, P. Maistri, " Detecting Faults in Integer and Finite Field Arithmetic Operations for Cryptography", *Proceedings of the Workshop on Fault Diagnosis and Tolerance in Cryptography*, Italy, June 2004.
URL: http://www.elet.polimi.it/res/FDTC04/Breveglieri.pdf