Highly Efficient Security Level Implementation in Radiation-Tolerance FPGA Using a Combination of AES Algorithm and Hamming Code: LST-SW Case

Assaad EL Makhloufi*, Samir EL Adib, and Naoufal Raissouni
Remote Sensing, Systems and Telecommunications (RST), University Abdelmalek Essaadi, Tetuan, Morocco
Email: assaadelmakhloufi@gmail.com* (A.EL.M.); {adibsamir; naoufal.raissouni.ensa}@gmail.com; (S.EL.A.); (N.R.)

***Abstract*—Nowadays, Satellite remote sensing and especially earth observation satellites require improvement when implementing an algorithm using reconfigurable hardware in terms of efficiency, performance, and security level. Furthermore, the transmission of the computed data from the earth observation satellite to the ground station is subject to potential attacks, and data can be corrupted during transmission. The land surface Temperature algorithm is an essential factor that can be extracted from the earth observation satellite. In this paper, we proposed a secure land surface temperature implementation by combining the advanced encryption algorithm and Hamming code implemented on radiation-tolerant Virtex-4QV field programmable gate array (FPGA). The iterative looping technique was used to implement the Advanced Encryption Standard (AES) algorithm with the aim of achieving high throughput and high security levels. The results showed that the proposed hardware implementation consumes 3319 Slices and 2 BRAMs while achieving higher throughput of 1854.82 Mbps. Moreover, we have evaluated the efficiency of the suggested cryptosystem by exploiting security measurement analysis such as Histogram, correlation coefficient, entropy, number of pixels change rate (NPCR), and unified averaged changed intensity (UACI). The proposed cryptosystem results prove the efficiency in terms of hardware consumed and high-security level. This implementation may be utilized, proposed, and even configurable for any earth observation satellites, including nano- and pico-CubeSats since we employed radiation-tolerant FPGA.**

***Index Terms*—AES, earth observation satellite, FPGA, Hamming distance, land surface temperature**

I. Introduction

Satellite Remote Sensing (SRS) provides a repetitive and synoptic view of the earth of great value in monitoring and analyzing various applications. This application can be used in the environmental monitoring, detection, and monitoring of global changes, exploration of non-renewable and renewable natural resources, meteorology, cartography, etc. [1].

The estimation of the surface temperature is one of the essential factors that can be computed using SRS [2, 3]. Actually, physical remote sensing methods are used to calculate the surface temperature from satellite measurements in the thermal infrared (TIR) window channels [4]. Different approaches for estimating the surface temperature have been introduced in the literature [5−9]. One of them is the land surface temperature-split window (LST-SW) algorithm proposed by Sobrino & Raissouni [10].

On-board computed land surface temperature (LST) uses a field programmable gate array (FPGA) circuit that can help to reduce the hardware resources. Data can be modified, reducing the amount of data to be transferred, and protects the hardware of the satellite against space radiation [11–13]. The implementation of the LST-SW algorithm based on radiation-tolerant FPGA has been introduced in our previous paper [14]. However, if there are attacks, this solution is not assuring the confidentiality of sensitive data, and the computed LST results could be corrupted during transmission.

During the transmission of these data to the earth station, they must be secured and encrypted against outside attacks because unauthorized incursions are possible, which means a satellite can be hacked [15, 16]. Therefore, the need for securing the transmission of the computed LST-SW algorithm from the satellite to the ground station is very important. As a solution, one of the powerful encryption algorithms is the advanced encryption standard (AES), this last, is a symmetric algorithm that can be applied to protect the sensitive and valuable data captured from the satellite and transmitted to the ground station [17−19]. In fact, the AES algorithm is adopted in many organizations and applications with the aim of securing communication links between the satellite and the ground station [20, 21]. Nevertheless, to reach a high level of protection, these organizations have decided to use hardware to implement AES algorithm; since it is quicker, more secure, and uses less power. Making it a more enticing alternative than software, which is more susceptible to outside assaults [22, 23]. The encryption algorithms typically protect potentially highly sensitive information between the satellite and the ground station. However, during transmission, data can

Manuscript received November 29, 2022; revised December 26, 2022; accepted December 30, 2022.
*Corresponding author: Assaad EL Makhloufi

be corrupted and lost due to the radiation in space, such as: a) total ionizing dose (TID), b) single event upset (SEU), and single-event effect (SEE). Consequently, appropriate precautions should be made to preserve the uplink and downlink from any threat, and protect the satellite's hardware from any type of radiation.

As a solution, implemented Hamming error correction and detection code, might detect and correct error during transmission. Richard Hamming created the hamming code in 1950 as a linear block error detection and correction system [24]. Furthermore, Hamming code is perfect, which indicates that there is no more compact code with the same correction capability for a particular code length. In this regard, its yield is optimal.

Consequently, a combination between the AES algorithm and Hamming code allowing single-error correction and single-error detection, has reduced threats from hackers and/or loss of sensitive data during transmission due to space radiation.

Several implementations were developed using the combination between the AES algorithm and Hamming code for space applications. However, in the field of remote sensing satellites and especially for securing the LST-SW algorithm using the combination, none in the literature has been done yet. Mohamed *et al.* in [25] described an FPGA implementation of a composite Hamming and AES algorithm for on-board satellite applications. The Chen *et al.* in [26] described their study on error detection and correction system (EDAC) schemes for memory in space applications using cascaded "Bose–Chaudhuri–Hocquenghem and cyclic redundancy check" codes and scrubbing approach. In [27], Hillier *et al*. presented error detection and correction on-board nanosatellites using Hamming codes. Hamming (16, 11, 4)$_2$ was developed using the very high-speed integrated circuit (VHSIC) hardware description language (VHDL) to guarantee single-error correction and double error detection. On the other hand, authors in [28] proposed a secure implementation improvement for the LST-SW algorithm. This improvement consisted on the use of the reconfigurable dynamic method for the secure LST-SW algorithm, and the main advantage of this implementation was the use of three different keys according to the state machine. However, this implementation does not include the Hamming code for detecting and correcting errors during data transmission to the ground station.

As the remote sensing satellite operates in a harsh radiation environment, the combination of the AES algorithm and Hamming code must be implemented in specific FPGAs [29]. Therefore, there is a special electronic circuit operating in space radiation environment. Because of its ability to perform in high-radiation environments, Virtex-4QV [30] FPGAs are ideal for space applications [31].

This paper describes a novel architecture using a combination between the AES algorithm and Hamming code for securing the LST-SW algorithm. The iterative looping technique has been choosing to implement the AES algorithm belong with Hamming code. The choice of the proposed technique is for the aim of reducing the design area and achieving the maximum throughput. Moreover, achieving a high level of security by combining the AES algorithm and Hamming code (15, 11, 4).

II. AES Algorithm

Joen Daemen and Vincent Rijmen, Belgian cryptographers, invented the Rijndael family of block ciphers [20]. It was presented as part of a competition sponsored by the National Institute of Standards and Technology to choose AES to replace the data encryption standard (DES). The competition was triumphed by Rijndael in 2001, with different key sizes been used in the three AES variations (128, 192, and 256 bits). Fig. 1 describes the process of the encryption and decryption of AES algorithm.

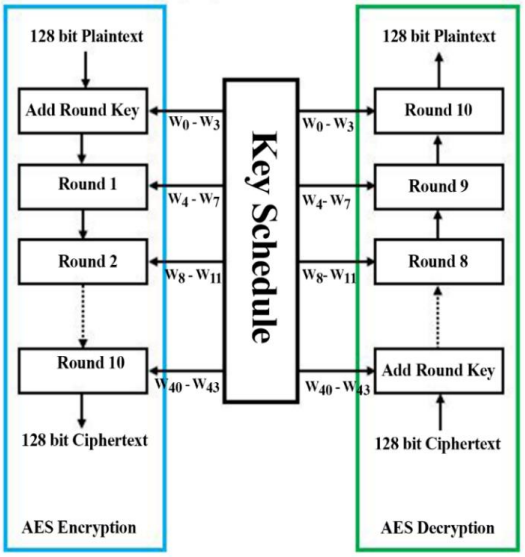

Fig. 1. Encryption/decryption process of AES algorithm.

Regular rounds in the encryption process include various phases: SubBytes, ShiftRows, MixColumns, and AddRoundKey. During the final round, the MixColumns step is skipped. For the decryption process, regular rounds are AddRoundKey, inverse SubBytes, inverse ShiftRows, and inverse MixColumns. The last round skips the inverse MixColumns transformation.

STEP 1: called SubBytes for the encryption process. The related substitution step during decryption is named InvSubBytes. The SubBytes phase entails dividing the input into bytes and sending each through a substitution box, often known as an S-Box. The inverse SubBytes transformation is applies the inverse S-box to each byte of the state, by computed the inverse of the affine transformation, pursued by the multiplicative inverse in Galois fields (GF).

STEP 2: called ShiftRows for the encryption process. The equivalent transformation is known as InvShiftRows during decryption. The ShiftRows operation shifts each of these rows to the left by a predetermined amount, with the row number beginning with zero. The top row is unaltered, the following row is relocated by one, and so on. The inverse ShiftRows operation is the same as the ShiftRows operation, except that the rotations are done to the right rather than the left.

STEP 3: During the forward step, MixColumns is used to mix up the bytes in each column independently. During decryption, the appropriate transformation is called InvMixColumns, which stands for inverse mix column transformation. The idea is to scramble the 128-bit input block. After 10 rounds of processing, the shift-rows and mix-column steps force each bit of the ciphertext to be dependent on every bit of the plaintext.

STEP 4: During the forward step, AddRoundKey is used to add the round key to the output of the preceding step. For inverse add round key transformation, the equivalent step during decryption is called InvAddRoundKey. S-box substitutions, word rotations, and XOR operations on the encryption key are all part of a round key generation (key expansion).

III. Hamming Code

The purpose of this SEU hardened technique is to identify and repair errors in flip-flops, registers, or memory and fix them when the stored value is used. To correct errors, additional logic structures are necessarily based on the value and type of stored cells in the circuit.

A. Overview of Hamming code: Hamming (7, 4) Case

Richard W. Hamming, an American mathematician, developed the Hamming Code technique for detecting and correcting errors [24]. Hamming introduced the concept of the number of locations in which two code-words differ and the number of modifications necessary to turn one code-word into another in a study published in 1950. It is currently known as the Hamming distance. This groundbreaking research not only solved a significant problem in telecommunications and computer science but also established a new field of study. Table I describe the features of a generic (*n*, *k*) Hamming code.

TABLE I: CHARACTERISTICS OF HAMMING (7, 4)

Code length	$n = 2^p - 1$
Number of information symbols	$k = 2^p - p - 1$
Number of parity symbols	$n - k = p$
Minimum distance	$d_{\min} = 3$
Error correcting capability	$t = 1$

The most basic binary Hamming code (7, 4) is obtained using the simplest configuration: p=3. The (7, 4) binary Hamming block encoder takes four-bit data blocks and adds three parity bits to each one, yielding seven-bit Hamming coded blocks.

B. Some Generator and Parity-Check Matrix Characteristics

The following property in Eq. (1) describes the generator matrix **G** and the parity-check matrix **H**.

$$\mathbf{G}\mathbf{H}^T = 0 \tag{1}$$

It should be noted that the generating matrix is in standard form, with the elements partitioned as:

$$\mathbf{G} = \left[\mathbf{I}_K \middle| \mathbf{P}\right] \tag{2}$$

where $\mathbf{I}_k$ is a $k \times k$ identity matrix and **P** is a $k \times (n-k)$ matrix which generates parity check digits. When **G** is a standard form matrix, the equivalent parity-check matrix **H** is simply calculated by:

$$\mathbf{H} = \left[-\mathbf{P}^T \middle| \mathbf{I}_{n-k}\right] \tag{3}$$

In Galois field, a number's negation is just its absolute value. As a result, the matrix for binary coding may be represented simply as:

$$\mathbf{H} = \left[\mathbf{P}^T \middle| \mathbf{I}_{n-k}\right] \tag{4}$$

C. Encoding Process

The parity bits are determined using modulo-2 additions from the following linearly independent equations given by:

$$\begin{aligned} P_1 &= D_1 \oplus D_2 \oplus D_3 \\ P_2 &= D_2 \oplus D_3 \oplus D_4 \\ P_3 &= D_1 \oplus D_3 \oplus D_4 \end{aligned} \tag{5}$$

A generator matrix **G** is implemented by the Hamming encoder. The generator matrix is easy to create using the linear equations stated in the equation above Eq. (5). The linear equations illustrate that the information bit D_1 has an impact on the parity computation at P_1 and P_3. Similarly, D_2 influences P_1 and P_2, D_3 influences P_1, P_2, and P_3, while D_4 influences P_2 and P_3.

$$\mathbf{G} = \begin{bmatrix} D_1 & D_2 & D_3 & D_4 & P_1 & P_2 & P_3 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{6}$$

The encoder receives a 4-bit message block, multiplies it with the generating matrix, and produces 7-bit codewords as matrix operations. It's worth noting that all of the operations (addition, multiplication, and so on) are performed in the modulo-2 domain. The Hamming encoder is responsible for generating the codeword from the message vector denoted by **M** and generator matrix **G**. The formula used to calculate the codeword is given as in Eq. (7):

$$\text{codeword}_{n\text{-bits}} = \text{mod}_2\left(\mathbf{M}_{k\text{-bits}}\mathbf{G}_{k\times n}\right) \tag{7}$$

D. Decoding Process – Syndrome Decoding

A parity check matrix **H** is created by combining the three check equations for the generator matrix (**G**) for the sample (7, 4) Hamming code that is provided in Eq. (8). The parity check matrix is important for error detection and correction on the receiver side.

$$\mathbf{H} = \begin{bmatrix} D_1 & D_2 & D_3 & D_4 & P_1 & P_2 & P_3 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

According to the parity-check theorem, every generator matrix **G** can have a parity-check matrix **H** that covers **G**'s null space. As an outcome, **c** will be orthogonal to every row of **H** if it is a suitable codeword as described in Eq. (9).

$$\mathbf{c}\mathbf{H}^T = 0 \tag{9}$$

As a result, if **H** is a codeword's parity-check matrix, a vector in the received code space is a valid codeword if and only if it fulfills the Eq. (9).

Consider a vector of received words **r**=**c**+**e**, where **c** is a valid codeword sent and e is the error that occurs by the channel. The syndrome for the received vector **r** is defined as the matrix product $\mathbf{r}\mathbf{H}^T$, which may be viewed as a linear transformation whose null space is c as shown in Eq. (10).

$$\mathrm{S} = \mathbf{r}\mathbf{H}^T = (\mathbf{c}+\mathrm{e})\mathbf{H}^T = \mathbf{c}\mathbf{H}^T + \mathbf{e}\mathbf{H}^T = 0 + \mathbf{e}\mathbf{H}^T = \mathbf{e}\mathbf{H}^T \tag{10}$$

As a result, the syndrome is unrelated to the transmitted codeword **c** and is exclusively caused by the error pattern. If two error vectors e and **e**' have the same syndrome, the error vectors must vary by a nonzero codeword.

$$S = \mathbf{e}\mathbf{H}^T = \mathbf{e}'\mathbf{H}^T = (\mathbf{e}-\mathbf{e}')\mathbf{H}^T = 0 \Rightarrow$$
$$(\mathbf{e}-\mathbf{e}')\mathbf{H}^T = 0 \Rightarrow (\mathbf{e}-\mathbf{e}') = \mathbf{c} \in \mathbf{C} \tag{11}$$

According to the equation above, the decoding may be done by calculating the received words syndrome, locating the associated error pattern, and then subtracting (or adding in the GF domain) the error pattern from the received word. This eliminates the need to save every vector as in a typical array decoding and significantly lowers the memory requirements for the decoder's implementation.

In general, we can calculate the syndrome from Eq. (12):

$$\text{syndrome} = \mathrm{mod}_2\left(\mathbf{C}_{n\text{-bits}}\mathbf{H}^T_{k\times n}\right) \tag{12}$$

E. Hamming (15, 11, 4)

A generator matrix (**G**) and parity-check matrix (**H**) are required for the encoding, decoding, and computation of the syndrome in order to use the Hamming (15, 11, 4).

The computations to construct the code (15,11) Hamming code are as follows, and the below diagrams show the systematic forms of the 4×15 parity-check matrix **H** and the 11×15 generator matrix **G**.

$$\mathbf{H} = \mathbf{I}_4\mathbf{Q} = \begin{bmatrix} 1&0&0&0&1&1&0&1&0&0&1&1&1&0&1 \\ 0&1&0&0&1&0&1&0&1&0&1&1&0&1&1 \\ 0&0&1&0&0&1&1&0&0&1&1&0&1&1&1 \\ 0&0&0&1&0&0&0&1&1&1&0&1&1&1&1 \end{bmatrix} \tag{13}$$

$$\mathbf{G} = \mathbf{Q}^T\mathbf{I}_{11} = \begin{bmatrix} 1&1&0&0&1&0&0&0&0&0&0&0&0&0&0 \\ 1&0&1&0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&1&1&0&0&0&1&0&0&0&0&0&0&0&0 \\ 1&0&0&1&0&0&0&1&0&0&0&0&0&0&0 \\ 0&1&0&1&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&1&1&0&0&0&0&0&1&0&0&0&0&0 \\ 1&1&1&0&0&0&0&0&0&0&1&0&0&0&0 \\ 1&1&0&1&0&0&0&0&0&0&0&1&0&0&0 \\ 1&0&1&1&0&0&0&0&0&0&0&0&1&0&0 \\ 0&1&1&1&0&0&0&0&0&0&0&0&0&1&0 \\ 1&1&1&1&0&0&0&0&0&0&0&0&0&0&1 \end{bmatrix} \tag{14}$$

Once **G** and **H** have been determined, the information data can be encoded and decoded in a way that allows for single-error correction and single-error detection. The data is encoded using Eq. (7).

IV. Proposed Hardware Architecture

The proposed Radiation-Tolerant FPGA implementation is focused on combining the AES algorithm using the iterative looping architecture with the hamming code to achieve a high-security level. The aim of the proposed design is to optimize the design area (hardware resources), maximize operating frequency, and achieve the maximum throughput and high security level to be adequate for secure mission in remote sensing satellite.

Fig. 2 describe the proposed architecture with the integration of AES algorithm and Hamming code for remote sensing satellite mission.

The following steps described how the proposed system securing the LST-SW algorithm:

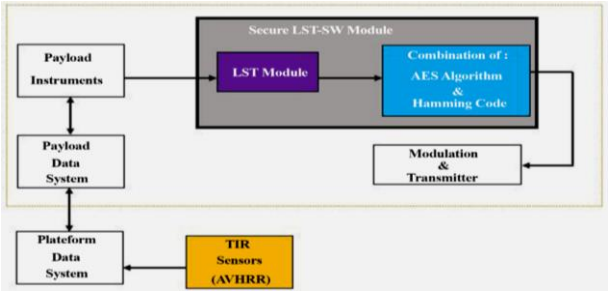

Fig. 2. Proposed AES/Hamming integration module in SRS.

Step 1. The satellite captures images (T4, T5, W, Epsilon) and gathers information from the Earth's surface and beyond using the AVHRR sensors.

Step 2. The first module is for computing the LST-SW algorithm.

Step 3. The result of the LST module goes to the second module which contains the combination of the AES algorithm and hamming code. After each transformation, we integrate the encoder and decoder.

Step 4. Send the computed LST-SW to the ground station with a high level of security and without any bit of error or corruption during transmission.

A. Proposed Combination Schema Between AES and Hamming Code

The focus of the iterative looping technique is on implementing only one round as a combinational processing element. Each step of the proposed architecture using the iterative looping architecture contains Hamming coding and hamming decoding to check if there is any error occurs in output of each step.

The output from every round is saved in a shared register, which also serves as the input for the subsequent round. The outcome from the previous clock cycle is therefore saved using the data register. As a result, 10 clock cycles must pass before starting the next LST-Pixel. Fig. 3 presents the processing diagram for the suggested Iterative Looping architecture.

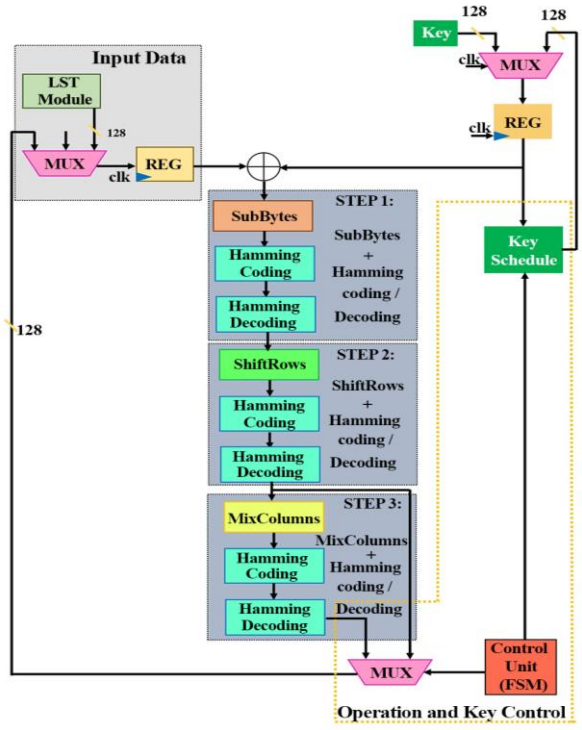

Fig. 3. Proposed AES/Hamming integration module based on the iterative looping architecture.

We developed a controller block (FSM control) to make sure that each block undergoes the transformations the right number of times and to avoid data collisions when a new block reaches the datapath because the datapath for the presented work has an iterative looping design. The final round also slightly deviates from the previous rounds by applying just three transformations as opposed to four due to the removal of the Mixcolumn layer, in addition to controlling hamming coding and decoding in each output of the AES algorithm. To detect this change, a control circuit is therefore required. We created the control module to adhere to the following principles:

- Follow each block as it passes along the datapath until it reaches the final add round key instance.
- Controlling the last round of the AES algorithm by omitting the mixcolumns transformation.
- Controlling the hamming coding and decoding in each output's transformation.
- Until a new block of data is prepared to enter the suggested architecture, the FSM is responsible for resetting the output of the initial add round key instance.
- Controlled the computed LST-SW pixels by sending one pixel in each round.

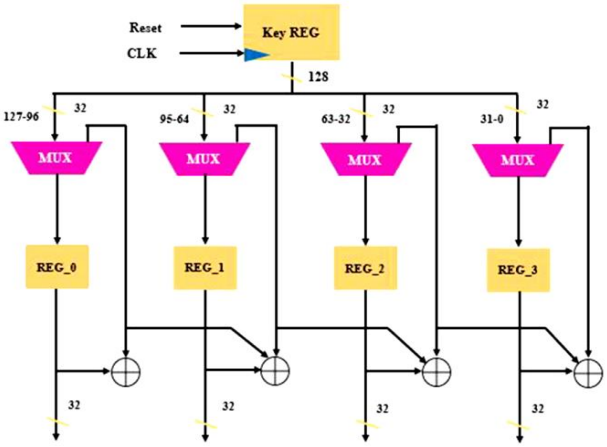

Fig. 4. Block diagram of key expansion.

The Key schedule unit is controlled by the FSM unit by sharing the key in each round as described in Fig. 4 which represents one round of the key schedule algorithm. This helps to generate round keys on the fly without the need of storing the keys which helps in saving space. This implementation technique is more efficient in space mission since no additional storage capacity is required to save the 10 precomputed Round Keys. Furthermore, the parallel computing adds no extra delay to the critical path or latency to the system.

The proposed design for secure LST-SW using the AES algorithm with detecting and correcting is utilizing the Hamming code from a pre-calculated Hamming code look up table at the finish of each transformation. The memory stores the pre-calculated output bits of each input byte in the state matrix as S-Box. The hamming code memory of the state matrix is anticipated after each output of transformation in AES. Meanwhile, the Hamming codes are computed again using the AES transformation output. As shown in Fig. 5, the expected and computed Hamming codes are compared to detect and correct the fault.

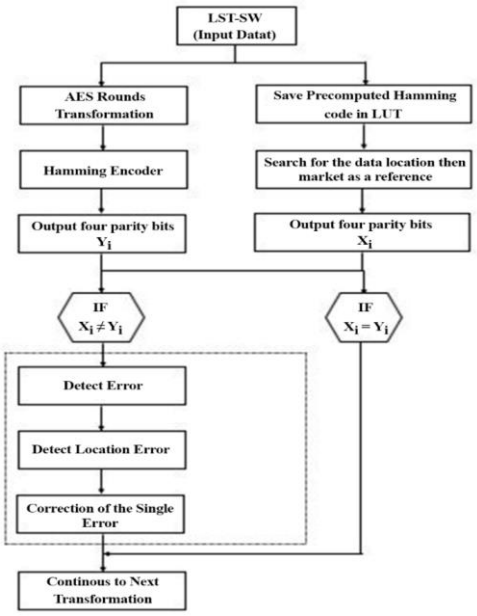

Fig. 5. Design flow chart for the Hamming code.

B. Internal Architecture of the Proposed Implementation

This section is used to analyze the proposed architecture using the Register Transfer Level (RTL) viewer for the general architecture (Top Level), the internal of each transformation, and the internal architecture of the hamming coding, and decoding. The RTL view of the top-level architecture is shown in the following Fig. 6.

The composition of top level is the same in the proposed architecture described in Fig. 3, which contains:

i) The four Inputs of the design T4, T5, W and Epsilon.
ii) LST_Part1 Module for the calculation of the first part of the algorithm.
iii) LST_Part2 Module for the calculation of the second part (Correction Part) of the algorithm.
iv) The four transformations of the AES Algorithm: sub_byte, shift_rows, mix_columns, and the add_round_key.
v) The controlled Unit (FSM)
vi) The outputs computed AES-LST after applying the combination of the AES and hamming code.

The following Figures shows the internal architecture of each transformation belonging to the connection signals. The three main steps are described as follows:

Steps 1: After calculating the first pixel of the LST-SW algorithm belonging to the initial round of the AES algorithm, the sub_byte transformation is performed, then the hamming coding and decoding are checked if any errors occur, if there is an error detected with the position of the bits, next the pre-computed table is corrected the error, then go through to next transformation. If there are no errors the output goes directly to the next transformation. Fig. 7 describe the sub-Byte transformation including the hamming code. The S-Box is repeated 16 times in Fig. 7.

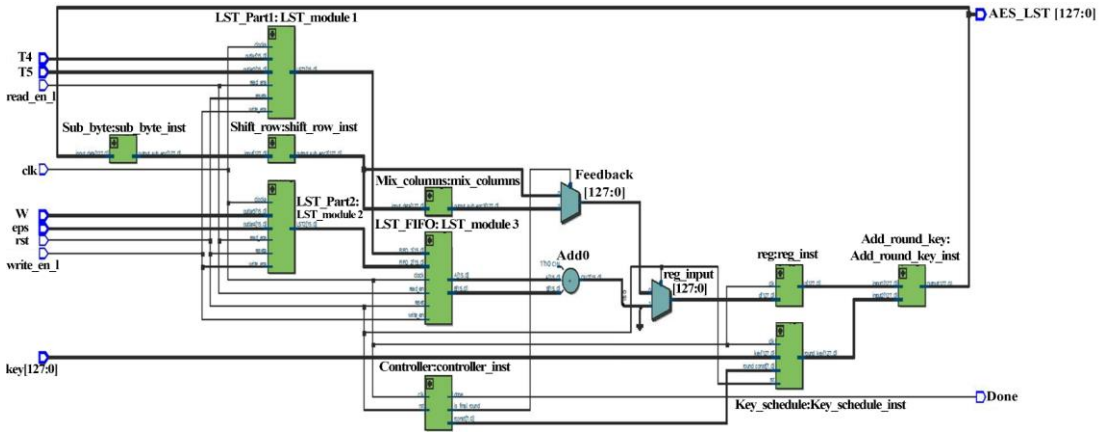

Fig. 6. RTL view of the proposed iterative looping architecture.

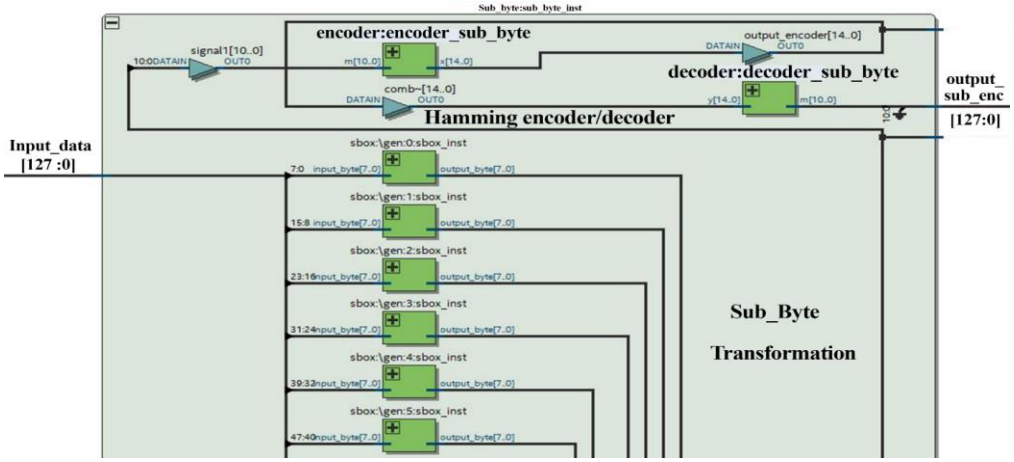

Fig. 7. RTL view of the SubByte transformation including the hamming encoder/decoder.

Step 2: The output of the sub_byte transformation goes to the next transformation of the shift_row to be performed. The output of the shift_row proceeds to the hamming coding and decoding to check if any errors occurring, if there is an error detected with the position of the bits, next the pre-computed table is corrected the error, then goes through to the next transformation as shown in Fig. 8. If there are no errors the output goes directly to the next transformation.

Step 3: The output of the shift_row transformation goes to the subsequent transformation of the mix_columns to be performed. The results of the mix_columns go to the hamming coding and decoding to check if any errors occurring, if there is an error detected with the position of the bits, next the pre-computed table is corrected the error, then goes through to the next transformation as described in Fig. 9. If there are no errors the output goes directly to the next transformation.

These steps are repeated in each computed pixels of the LST-SW algorithm.

The Hamming encoder and decoder modules perform on each output transformation. Accordingly, this module is repeated 6 times after each round of the AES algorithm. The 6-time is repeated again 10 times in the hall process of the AES transformation rounds. As a result, SEUs are recognized and repaired before proceeding to the next AES transformation. As a consequence, the AES and Hamming processes can operate without interfering with one another.

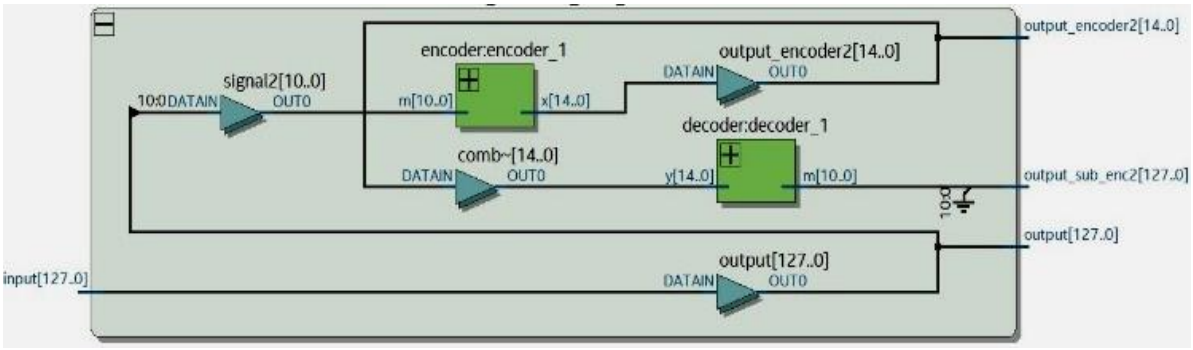

Fig. 8. RTL view of the ShiftRows transformation including the hamming encoder/decoder.

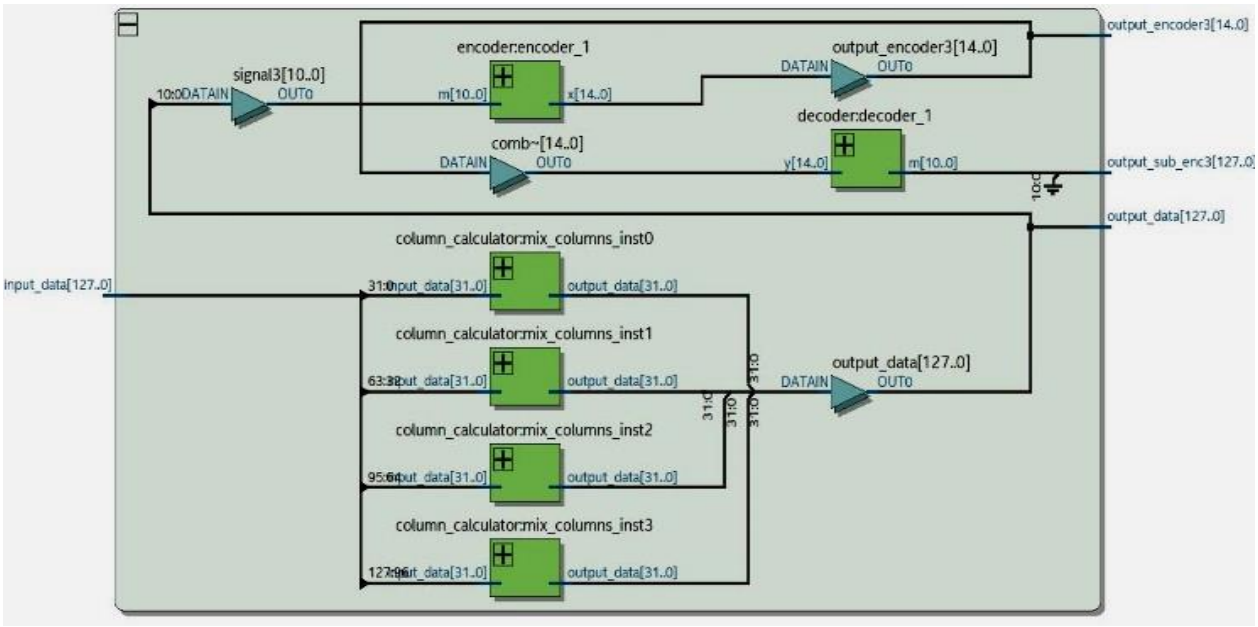

Fig. 9. RTL view of the MixColumns transformation including the hamming encoder/decoder.

TABLE II: HARDWARE RESOURCES CONSUMED IN THE PROPOSED IMPLEMENTATION

Target FPGA Device utilization	Virtex-4QV		
	Used	Total	%
Slice	3319	24576	13
Slice LUTs	3090	49152	6
DSP48Es	18	512	3
Block RAM / FIFO	2	60	1
Bonded IOBs	325	640	50
Maximum Frequency	173.89 MHz		
Cycles	12		
Throughput	1854.82 Mbps		

V. HARDWARE AND SIMULATION RESULTS

A. Hardware Implementation Results

The combination of AES algorithm and Hamming code is programmed using VHDL and synthesized using ISE 14.7 tools, conducted on the space-grade Virtex-4QV XQR4VSX55. Table II provide the resources and synthesis results used for the hardware implementation for encryption 128 bits data with AES-128 Key.

The proposed implementation includes all the components: The computed LST-SW algorithm, AES encryption process which contains the integration of Hamming coding and decoding in each transformation has consumed 3319 Slice register, 3090 Slice LUTs, 18 of DSP48Es Blocks, and just 2 block RAMs. Furthermore, as seen in Table II, the proposed implementation achieves a higher throughput of 1854.82 Mbps with a maximum frequency of 173.89 MHz.

B. Simulation Results

Simulation has now become rigorous and saves much time compared to a tune-up performed entirely on the hardware. In this work, to facilitate the design simulation, we created two files:

- Testbench of the top module as tb_LST-SW-AES. Vhd
- Testbench of the SubByte transformation including
- Hamming coding/decoding as LST-SW-AES_SubByte_Ham_cod_deco. Vhd,

Fig. 10 illustrate the simulation result for the top module (tb_LST-SW-AES. Vhd). This proposed system is controlled by CLK, reset_l, read_en_l, and write_en_l. Furthermore, the simulation shows all inputs and the output LST-SW of the complete system. Four inputs are used: T4, T5, W, eps, and the key used.

The following simulations proved the functionality of the combination of AES steps transformation and Hamming (15, 11, 4). Fig. 11 show the simulation result of the SubByte transformation including Hamming coding/decoding. The following should be noted:

- Input_data [127:0]: The input of the SubByte transformation
- Output_data [127:0]: The results of the SubByte transformation
- Output_Encoder[14:0] : The data has been computed in the hamming coder
- Output_Sub_enc [127:0]: Output_Encoder is sent to be decoded, then converted to 128 bits to be used as input for the next transformation.
- Error_detect: no Error was detected during the combination of AES algorithm and Hamming coder/decoder.

To test the robustness of the proposed implementation, we have injected fault bits in the subByte transformation. The following Fig. 12 shows that the signal of error_detect is set to 1 which means that decoder has detected an error.

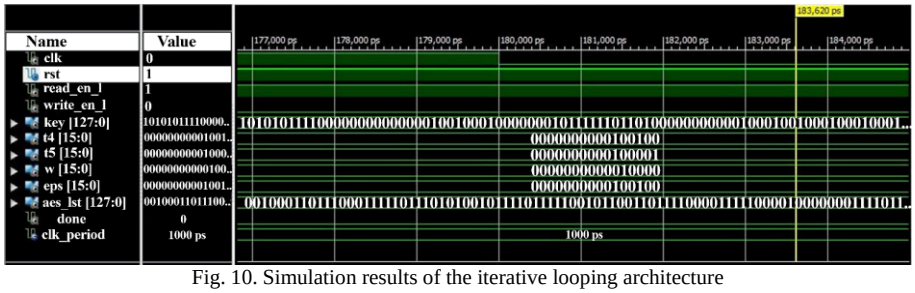

Fig. 10. Simulation results of the iterative looping architecture

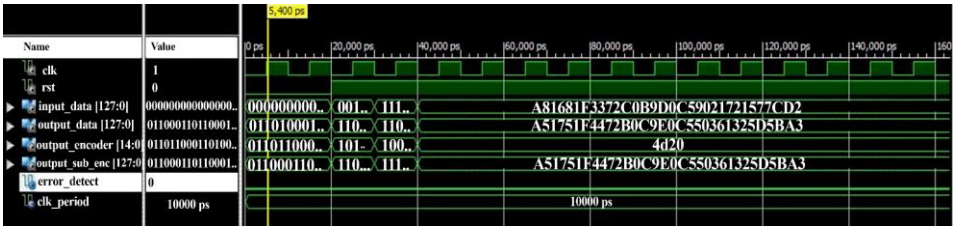

Fig. 11. Simulation results of the SubByte transformation including the hamming encoder/decoder without error_detect.

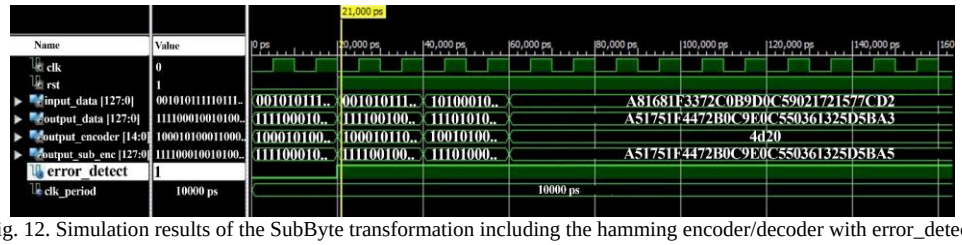

Fig. 12. Simulation results of the SubByte transformation including the hamming encoder/decoder with error_detect.

Let take an example of the computed LST-SW as input_data which is used in this simulation:

- Input_data:“01101111111111111100100010111111010110010010000010100111100111010110000001001010101101010000111011110111011101000000001 01111111 which equal in Hexadecimal: "A81681F3372C0B9D0C59021721577CD2"
- Output_data: is the computed subByte transformation equal in Hexadecimal to : “A51751F4472B0C9E0C550361325D5BA3”
- Output_encoder: “1001101000100000” equal in Hexadecimal to: “4d20”
- Output_sub_ecr: is the output of the decoder after decoding the previous input is equal in Hexadecimal to “A51751F4472B0C9E0C550361325D5BA5”

As seen in the simulation error_detect is equal to 1

which means there is an error detected. In fact, the Hamming decoder first detects the error, locates it using the even parity checking technique, and corrects it using the Hamming detection method, which uses the even parity check to determine the error's position. The decoder corrects the erroneous bit after identifying its position by changing one for zero and zero for one.

VI. Security Analysis

A good encryption system should resist known attacks, so using different evaluation measures to verify the security and efficiency of the presented implementation. We will present the most important as: statistical tests such as (histogram, correlation, entropy) and differential tests such as: Unified Average Changing Intensity (UACI), and Number of Pixel Change Rate (NPCR) [32].

A. Histogram Analysis

The histogram is a graphic representation which allows to know the distribution of the luminous intensities of the pixels. Therefore, the histogram of the original image should differ from the histogram of the encrypted image.

The results in Fig. 13 show that the histograms of the encrypted LST-SW are uniform compared to the histogram of the computed LST-SW which is non-uniform. As a result, the suggested histograms meet the histogram's property. Therefore, the attacker cannot extract the information from the histogram of the encrypted LST-SW.

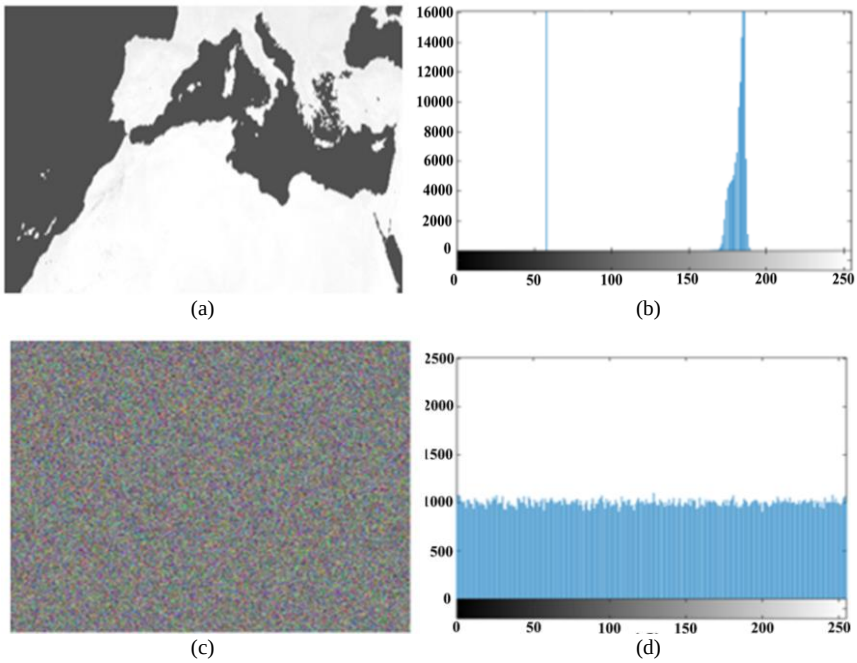

Fig. 13. (a) Computed LST-SW (Original), (b) Histogram of the LST-SW, (c) Encrypted LST-SW, (d) Histogram of the encrypted LST-SW

B. Correlation Coefficient Analysis

Adjacent pixels in an original image are strongly correlated, but in the encrypted become weakly correlated. Computing the correlation coefficient between neighbouring pixels gives us an idea of the ability of our encryption algorithm to resist attacks. The correlation coefficient formula is defined in [28].

To study the correlation, we have calculated the correlation coefficients in the three directions (vertical, horizontal and diagonal) between the computed LST-SW and the encrypted LST-SW. The Fig. 14 above shows the curves of the correlations between the two images. We have selected 3000 randomly nearby pixels from the two images to be used for the correlation coefficient analysis.

Fig. 14 (a), (c), and (e) depict a strong correlation between nearby pixels. However, in Fig. 14 (b), (d), and (f) the connection between neighboring pixels for an encrypted LST-SW is weak. Therefore, the proposed secure implementation is highly efficient in terms of correlation coefficient.

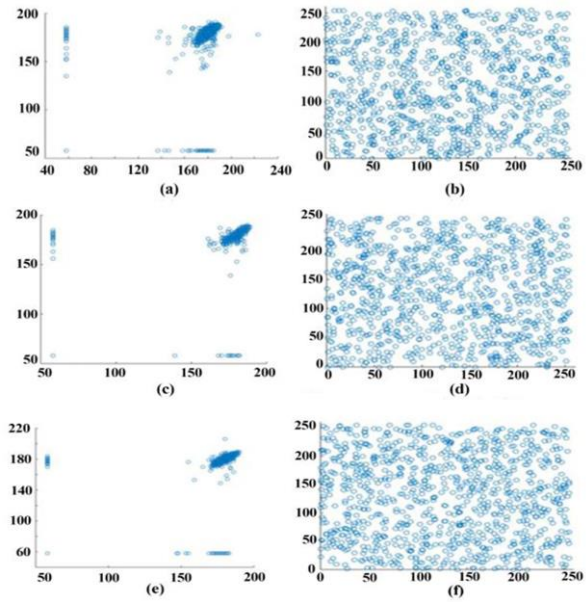

Fig. 14. (a, c, e) the horizontal, vertical and diagonal correlations of the computed LST-SW, while (b, d, f) is for the encrypted LST-SW

The calculation results in the Table III below show that the encrypted LST-SW values of the correlation is very low in the three levels (diagonal, horizontal, vertical) compared to the correlation of the original image.

TABLE III: CORRELATION COEFFICIENT ANALYSIS

LST-SW	Vertical Correlation	Horizontal Correlation	Diagonal Correlation
Computed	0.9788	0.9712	0.9602
Encrypted	-0.0120	0.0034	-0.0123

Therefore, the proposed secure implementation is highly efficient in terms of the correlation coefficient.

C. Entropy and Sensitivity Analysis

1) Entropy

Entropy indicates the level of uncertainty in communication system. The entropy H(x) of any data is computed as follows:

$$\text{Entropy} = \sum_{i=0}^{n} P_i \log_2 P_i \qquad (15)$$

The entropy value must be very close to 8, because if the entropy is less than 8, there are degrees of predictability.

2) Sensitivity Analysis

Two typical parameters may be used to examine the impact of a single pixel change on the entire picture encrypted by any encryption algorithm: NPCR and UACI. The NPCR calculates the proportion of distinct pixels in the two images. In contrast, the UACI calculates the average intensity of the discrepancies before and after encryption. The equations of NPCR and UACI is described in [28].

TABLE IV: SENSITIVITY AND ENTROPY ANALYSIS

Image	**Size**	**NPCR (%)**	**NPCR (%)**	**Entropy**
AES/Hamming LST-SW	316×695	99.76823	33.35687	7.9997

The sensitivity and entropy analysis results are reported in Table IV. According to the NPCR and UACI scores, the suggested approach is exceptionally resistant to a wide range of threats. Because the NPCR score is around 100% and the UACI is near the value of 33.33%. Moreover, the entropy in Table IV is 7.9997, which is near to the ideal value of 8 and indicates that the proposed encryption approach properly randomized the pixels in the encrypted LST-SW.

VII. PERFORMANCE COMPARISON TO THE EXISTING WORK IN TERM OF HARDWARE AND SECURITY ANALYSIS

In this section, we evaluate the proposed and the improvement implementation of the combination of AES algorithm and hamming code to the existing work.

As can be observed from Table V, the proposed implementation yields better performance in terms of hardware metrics and throughput comparing the existing implementation. In fact, the proposed implementation which combines the AES algorithm and Hamming code consumes fewer block slice and just 2 blocks of RAM compared to the existing work even without the integration of hamming code, and achieve the highest throughput metric of 1854.82 Mbits/s with a clock frequency of 173.89 MHz. Moreover, the presented implementation is the only implementation that used the combination of the AES algorithm and hamming code, which is lead to a high-security level for securing the computed LST and for detecting and correcting errors during the transmission to the ground station. On the other hand, the work in [28] has the advantage of the use of three different key lengths (128, 192, 256) bits, which lead to a high-security level. However, this implementation in [28] does not address the problem of the loss and corrupted data during the transmission, and consumes more resources. Therefore, the proposed implementation is the best one in terms of area utilization and achieved a high throughput sufficient for satellite remote sensing and especially in the case of the LST-SW algorithm.

Table VI illustrates a comparison in terms of security analysis between the proposed implementation and existing work.

As indicated in Table VI, the correlation coefficients in black are near 0, and are smaller than that in the existing work. Moreover, the proposed system has the best security metric of NPCR and UACI. However, in the entropy metric, the other implementation has the best value, even if they are very close.

TABLE V: PERFORMANCE COMPARISON OF THE PRESENTED IMPLEMENTATION TO THE EXISTING WORK

Design	Device	Slices	Block RAMs	Fmax (MHz)	Throughput (Mbits/s)	Radiation-Tolerant	Security Level	Bit errors (Hamming code)
[11]	Virtex-5 LX50	1169	3	83.33	118.700	✗	✗	✗
[14]	Virtex-4 QVSX55	1124	–	190.48	435.392	✓	✗	✗
[28]	Virtex-4 QVSX55	4089	4	145.36	907.644	✓	✓	✗
Proposed	Virtex-4 QVSX55	3319	2	173.89	1854.82	✓	✓	✓

TABLE VI: PERFORMANCE COMPARISON OF THE PRESENTED IMPLEMENTATION TO THE EXISTING WORK IN TERMS OF SECURITY ANALYSIS

Design	Device	Correlation			Entropy	NPCR	UACI
		Vertical Correlation	Horizontal Correlation	Diagonal Correlation			
[28]	Virtex-4 QVSX55	−0.0280	0.0125	−0.0265	**7.9999**	99.6093	33.4635
Proposed	Virtex-4 QVSX55	**−0.0120**	**0.0034**	**−0.0123**	7.9997	**99.76823**	**33.35687**

VIII. Conclusion

In this research article, we introduced a novel implementation technique for secure LST algorithm that is based on merging between AES algorithm and hamming code. The iterative technique was used to implement AES algorithm. The hamming coding/decoding was insert in each transformation step of AES algorithm. We have utilized Virtex-4QV XQR4VSX55 FPGA in this implementation because its tolerance to radiation and appropriate for the earth observation satellite mission. The results depicted that the presented implementation consumes 3319 Slices and 2 BRAMs while achieving higher throughput of 1854.82 Mbps. Moreover, we have evaluated the performance of the encryption method using security measurement analyses such as Histogram, Correlation Coefficient, Entropy, NPCR, and UACI. Finally, we have compared the hardware results and security analysis metrics with the existing work, and the comparison revealed the efficiency in terms of hardware and a high level of security.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

The paper conceptualization, methodology, software, validation, implementation, and writing were done by Assaad EL Makhloufi. The supervision, project administration, review, editing, visualization, and analyzed the data were done by Samir EL Adib, and Naoufal Raissouni. All authors approved the final version.

Funding

This work was supported in part by the Presidency University Abdelmalek Essaadi (UAE/ENSATE-2020/21) grants for "Land Cover Dynamic Monitoring in Tangier-Tetuan-Al Hoceima Region in the Context of the Global Warming: Earth Observation Satellites and High-Performance Computing Contribution".

References

[1] P. S. Roy, M. D. Behera, and S. K. Srivastav, "Satellite remote sensing: Sensors, applications and techniques," *Proceedings of the National Academy of Sciences, India Section A: Physical Sciences*, vol. 87, no. 4, pp. 465–472, Dec. 2017.

[2] V. García-Santos, J. Cuxart, D. Martínez-Villagrasa, M. A. Jiménez, and G. Simó, "Comparison of three methods for estimating land surface temperature from landsat 8-TIRS sensor data," *Remote Sensing*, vol. 10, no. 9, #1450, Sep. 2018.

[3] E. Mustafa, Y. Co, G. Lui *et al*., "Study for predicting land surface temperature (LST) using landsat data: A comparison of four algorithms," *Advances in Civil Engineering*, March. 2020, doi: 10.1155/2020/7363546.

[4] J. A. Sobrino and J. C. Jiménez-Muñoz, "Land surface temperature retrieval from thermal infrared data: An assessment in the context of the surface processes and ecosystem changes through response analysis (SPECTRA) mission," *Journal of Geophysical Research: Atmospheres*, vol. 110, 2005, doi: 10.1029/2004JD005588.

[5] A. Benmecheta, A. Abdellaoui, and A. Hamou, "A comparative study of land surface temperature retrieval methods from remote sensing data," *Canadian Journal of Remote Sensing*, vol. 39, no. 1, pp. 59–73, Jan. 2013.

[6] Z. Wan and J. Dozier, "A generalized split-window algorithm for retrieving land-surface temperature from space," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 34, no. 4, pp. 892–905, Jul. 1996.

[7] Z. Qin, A. Karnieli, and P. Berliner, "A mono-window algorithm for retrieving land surface temperature from Landsat TM data and its application to the Israel-Egypt border region," *International Journal of Remote Sensing*, vol. 22, no. 18, pp. 3719–3746, Jan. 2001.

[8] J. A. Sobrino, N. Raissouni, J. Simarro, F. Nerry, and F. Petitcolin, "Atmospheric water vapor content over land surfaces derived from the AVHRR data: application to the Iberian Peninsula," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 37, no. 3, pp. 1425–1434, May 1999.

[9] B. H. Tang, K. Shao, Z. L. Li, H. Wu, F. Nerry, and G. Zhou, "Estimation and validation of land surface temperatures from chinese second-generation polar-orbit FY-3A VIRR data," *Remote Sensing*, vol. 7, no. 3, pp. 3250–3273, 2015.

[10] J. A. Sobrino and N. Raissouni, "Toward remote sensing methods for land cover dynamic monitoring: Application to Morocco," *International Journal of Remote Sensing*, vol. 21, no. 2, pp. 353–366, Jan. 2000.

[11] N. Raissouni, S. El Adib, J. A. Sobrino, *et al*., "Towards LST split-window algorithm FPGA implementation for CubeSats on-board computations purposes," *International Journal of Remote Sensing*, vol. 40, no. 5–6, pp. 2435–2450, Mar. 2019.

[12] C. He, H. Fu, J. Zheng, *et al*., "A real-time tree crown detection approach for large-scale remote sensing images on FPGAs," *Remote Sensing*, vol. 11, no. 9, #1025, Jan. 2019.

[13] C. Gonzalez, D. Mozos, J. Resano, and A. Plaza, "FPGA implementation of the N-FINDR algorithm for remotely sensed hyperspectral image analysis," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 50, no. 2, pp. 374–388, Feb. 2012.

[14] A. El Makhloufi, N. Chekroun, N. Tagmouti, S. El Adib, and N. Raissouni, "Improvements in space radiation-tolerant FPGA implementation of land surface temperature-split window algorithm," *International Journal of Electrical and Computer Engineering*, vol. 11, no. 5, pp. 3844-3854, Oct. 2021.

[15] H. Chen, B. Zhai, J. Wu, C. Du, and J. Li, "A satellite observation data transmission scheduling algorithm oriented to data topics," *International Journal of Aerospace Engineering*, vol. 2020, Jul. 2020, doi: 10.1155/2020/2180674.

[16] P. Tedeschi, S. Sciancalepore, and R. Di Pietro, "Satellite-based communications security: A survey of threats, solutions, and research challenges," *Computer Networks*, vol. 216, Oct. 2022, doi: 10.1016/j.comnet.2022.109246.

[17] T. Vladimirova, R. Banu, and M. Sweeting, "On-board security services in small satellites," presented at MAPLD International Conference, Washington, 2005.

[18] O. Lo, W. J. Buchanan, and D. Carson, "Power analysis attacks on the AES-128 S-box using differential power analysis (DPA) and correlation power analysis (CPA)," *Journal of Cyber Security Technology*, vol. 1, no. 2, pp. 88–107, Apr. 2017.

[19] M. J. Dworkin, E. Barker, J. Nechvatal, *et al*., "Advanced encryption standard (AES)," *Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, Gaithersburg*, pp. 1-52, Nov. 2001. doi: https://doi.org/10.6028/NIST.FIPS.197

[20] J. Daemen and V. Rijmen, "The design of Rijndael: AES - The advanced encryption standard," *Berlin Heidelberg: Springer-Verlag*, 2002. doi: 10.1007/978-3-662-04722-4.

[21] J. Daemen and V. Rijmen, "The block cipher Rijndael," in *Smart Card Research and Applications*, Berlin, Heidelberg, 2000, pp. 277–284. doi: 10.1007/10721064_26.

[22] A. El Makhloufi, N. Chekroun, N. Tagmouti, S. E. Adib, J. A. Sobrino, and N. Raissouni, "AES/FPGA encryption module integration for satellite remote sensing systems: LST-SW case," in *Proc. of 2020 3rd International Conference on Advanced Communication Technologies and Networking*, Sep. 2020, doi: 10.1109/CommNet49926.2020.9199644.

[23] A. El Makhloufi, N. Chekroun, N. Tagmouti, S. El Adib, J. A. Sobrino, and N. Raissouni, "FPGA/LST-SW encryption module

implementation for satellite remote sensing secure systems," in *Proc. of 2020 Fourth International Conference On Intelligent Computing in Data Sciences (ICDS)*, Oct. 2020, doi: 10.1109/ICDS50568.2020.9268739.
[24] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, Apr. 1950.
[25] S. Mohamed, K. A. Shehata, H. H. Issa, and N. H. Shaker, "FPGA implementation of a combined hamming — AES error tolerant algorithm for on board satellite," in *Proc. of 2015 World Congress on Information Technology and Computer Applications*, Jun. 2015, doi: 10.1109/WCITCA.2015.7367028.
[26] M. Chen, C. Guo, L. Chen, *et al*., "Research on EDAC schemes for memory in space applications," *Electronics*, vol. 10, no. 5, Jan. 2021, doi: 10.3390/electronics10050533.
[27] C. Hillier and V. Balyan, "Error detection and correction on-board nanosatellites using hamming codes," *Journal of Electrical and Computer Engineering*, vol. 2019, #e3905094, Feb. 2019, doi: 10.1155/2019/3905094.
[28] A. EL Makhloufi, N. Chekroun, S. E. Adib, and N. Raissouni, "Improved security approach based on AES algorithm for LST retrieval using satellite imagery in radiation-tolerant FPGAs," *IJERTCS*, vol. 13, no. 1, pp. 1–17, Jan. 2022.
[29] L. Rockett, D. Patel, S. Danziger, B. Cronquist, and J. J. Wang, "Radiation hardened FPGA technology for space applications," in *Proc. of 2007 IEEE Aerospace Conference*, Mar. 2007, doi: 10.1109/AERO.2007.353098.
[30] Space-grade Virtex-4QV FPGA. [Online]. Available: *Xilinx*. https://www.xilinx.com/products/silicon-devices/fpga/virtex-4qv.html
[31] M. J. Wirthlin, "FPGAs operating in a radiation environment: lessons learned from FPGAs in space," *J. Inst.*, vol. 8, no. 2, Feb. 2013, doi: 10.1088/1748-0221/8/02/C02020.
[32] C. H. Yang and Y. S. Chien, "FPGA implementation and design of a hybrid chaos-AES color image encryption algorithm," *Symmetry*, vol. 12, no. 2, Feb. 2020, doi: 10.3390/sym12020189.

Assaad EL Makhloufi received the M.S. degree in telecommunications systems and networks, in 2017 and a Ph.D. degree in embedded systems and telecommunications, in 2022, from the National Engineering School for Applied Sciences, Tetuan, Morocco at the University Abdelmalek Essaadi. Actually, he is an automotive and aeronautical industry engineer at ALTEN. He is a member of the Remote Sensing, Systems and Telecommunications Research Unit. His research interest focuses on the integration of remote sensing algorithms in embedded systems, applications of reconfigurable hardware to cryptography, mainly for nano- and pico- remote sensing satellites.

Samir EL Adib received the degree in informatics and electronics and M.S. degree in automatic and data processing, and Ph.D. degree in electronics from the University Abdelmalek Essaadi, Tetuan, Morocco, in 2004, 2006 and 2013, respectively. He is associate professor of embedded systems, cryptography and electronics at the National Engineering School for Applied Sciences, Tetuan, Morocco. He is heading the Remote Sensing, Systems and Telecommunications Research Unit. His main research interests are FPGAs in custom-computing applications, and more concretely, applications of reconfigurable hardware to cryptography for nano- and pico- satellites in remote sensing.

Naoufal Raissouni received the M.S., and Ph.D. degrees in applied physics (Satellite Remote Sensing) from the University of Valencia, Spain, in 1997, and 1999, respectively. He is full professor of signal processing and remote sensing at the National Engineering School for Applied Sciences in Tetuan, Morocco. He is heading the Remote Sensing & Geographic Information Systems Unit and a member of the Remote Sensing, Systems and Telecommunications Research Unit. His research interests include atmospheric correction in visible and infrared domains, the retrieval of emissivity and surface temperature from satellite data, embedded remote sensing applications, nano- and pico- satellites, and remote sensing environment applications.