A Scalable and Configurable Architecture for Hardware Authenticated Encryption Modules Compliant with the CCSDS Security Specifications

1st Luca Crocetti
Dept. of Information Engineering
University of Pisa
Pisa, Italy
luca.crocetti@unipi.it

2nd Luca Fanucci
Dept. of Information Engineering
University of Pisa
Pisa, Italy
luca.fanucci@unipi.it

3rd Luca Baldanzi
IngeniArs S.r.l.
Pisa, Italy
luca.baldanzi@ingeniars.com

4th Francesco Falaschi
IngeniArs S.r.l.
Pisa, Italy
francesco.falaschi@ingeniars.com

***Abstract*—Cybersecurity is one of the most challenging aspects in the modern Information and Communications Technology (ICT) era, including space applications. The Consultative Committee for Space Data Systems (CCSDS) is issuing and updating reports and standards to address this problem in the space sector. It defined the format of secure frames to protect data with different security features and the corresponding cryptographic algorithms to be applied. Among them, the Galois/Counter Mode (GCM) of the Advanced Encryption Standard (AES) is the only one that constitutes a comprehensive solution for the simultaneous confidentiality, integrity, and authentication of data (i.e. authenticated encryption). In this work, we present a configurable and scalable architecture for implementing hardware AES-GCM modules aimed at securing space applications compliant with the CCSDS specifications. The proposed architecture was designed using SystemVerilog and characterized in terms of trade-offs between resource utilization and maximum frequency by analyzing the implementation results on a space-grade KU060 FPGA. Indeed, the configurability at the synthesis level of the proposed architecture supports different approaches that can be exploited to find the most efficient solution for the target application. For this reason, we present two use cases for the integration of the proposed security module in a transmitter for CCSDS-compliant telemetry (TM) applications. The corresponding results confirm the adaptability of our solution in different application scenarios thanks to its configurability. In addition, they show that our module offers long-term protection in terms of classical and post-quantum security for modern space applications with a minimum resource cost of 672 Configurable Logic Blocks (CLBs), i.e. 1.6% of the FPGA resources.**

***Index Terms*—Space security, CCSDS, SDLS, Authenticated Encryption, AES-GCM, space-grade FPGA, KU060, Telemetry**

I. Introduction

Last years saw a growing number of incidents in satellites due to cyberattacks and violations of security [1] as shown in Fig. 1. This was the result of an increasing innovation in malicious attacks according to the increasing innovation of the New Space era started in the early 2000s. The widespread adoption of the Internet in organizational IT practices offered increasingly wider attack surfaces that also allowed indirect attacks exploiting the satellite Internet connections to find valid subscriber IP addresses to infect other servers. Indeed, the authors of [2] demonstrated how cyberattacks can undermine key features such as limited accessibility, attributable norms, and environmental interdependence in the space domain. In addition, actual statistics indicate that this trend could be carried on in future decades [1].

To address this aspect, the Consultative Committee for Space Data Systems (CCSDS) is issuing and updating a series of reports and standards for the definition of se-

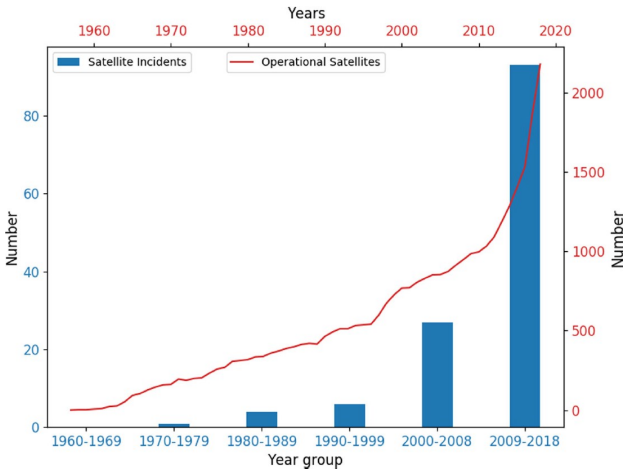

Fig. 1. Report on the space sector security [1]: number of operational satellites (red line) vs. number of satellite incidents due to security attacks (blue bars).

This work was partially funded by the Italian Ministry for Universities and Research (MUR) and by IngeniArs S.r.l.

curity requirements that can be applied, with appropriate differences, to both the satellite-to-ground (downlink) and ground-to-satellite (uplink) communication links. Indeed, remote configurations/controls sent to spacecraft through the uplink as telecommand (TC) packets need protection against unauthorized tampering or forgery of the commands to prevent damage to the payload, loss of vehicle control, etc. Therefore, authentication and data integrity mechanisms are requested. The downlink instead is used to transfer the scientific payload data generated by the instruments from spacecraft to ground stations as telemetry (TM) packets. They constitute a large amount of confidential information so, in this case, also the confidentiality protection must be included [1], [3]. In this context, the CCSDS defined specific security protocols relying on secure frames and the corresponding cryptographic algorithms to build such secure frames. Among the cryptographic algorithms approved by the CCSDS, the Galois/Counter Mode (GCM) of the Advanced Encryption Standard (AES) is the only one that supports a comprehensive solution for the simultaneous confidentiality, integrity, and authentication of data (i.e. authenticated encryption). For this reason, in this work, we present a scalable and reconfigurable architecture at the synthesis level that can be used to implement hardware AES-GCM modules to satisfy the requirements of the target application while minimizing resource utilization.

The remainder of the paper is organized as follows: Section II describes the security specifications from the CCSDS by focusing on the secure frame format and the associated cryptographic algorithms; Section III illustrates the main outline and the key points of the configurable architecture we propose to implement hardware modules offering authenticated encryption in space communications; Section IV presents an implementation-aware analysis of the trade-offs in terms of resource consumption, latency, maximum supported frequency, and throughput; Section V presents two use cases for the integration of the proposed module into a transmitter compliant with the CCSDS specifications for TM applications; finally, Section VI summarizes the main outcomes of this work and illustrates the follow-ups of our research activity.

II. CCSDS Security Protocol and Algorithms

Based on the main security threats reported in [4], the CCSDS emanated two standards for the security requirements of space applications: the standard 355.0-B [5] and the standard 352.0-B [6]. The former specifies the methods for applying cybersecurity services at the Space Data Link (SDL) layer. In particular, it defines a new Secure Frame format (Fig. 2) that includes the additional fields Security Header and Security Trailer compared to the non-secure Transfer Frame. The first field provides information on the security service (cryptographic algorithm, key identifier, etc.), whereas the second field constitutes the integrity proof of the frame. The content of Frame Data can be modified (or not) according to the security service. In the case of confidentiality protection, the data are replaced with their corresponding encrypted version (ciphertext), whereas they are left unchanged otherwise,

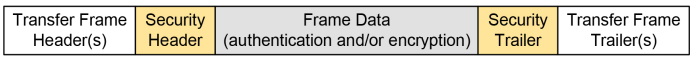

Fig. 2. Secure frame format defined by the CCSDS.

but anyway protected by authentication. The Secure Frame constitutes the data unit of the secure protocols that are defined at the same communication layer (Data Link) and takes the name of Space Data Link Secure (SDLS) layer. The SDLS layer operates transparently compared to the SDL layer. Indeed, from the SDL layer point of view, the sequence of Security Header, Frame Data, and Security Trailer is interpreted as the typical non-secure Frame Data field. Then, in case the SDLS protocol is applied, dedicated mechanisms encode (decode) the frame before (after) the SDL layer upon transmission (reception) of frames.

The CCSDS 352.0-B standard, on the other hand, specifies which cryptographic algorithms are to be used to apply security services. Hash-based Message Authentication Code (HMAC) [7], Cipher-based Message Authentication Code (CMAC) [8], and Galois Message Authentication Code (GMAC) [9] are authentication techniques that use a secret key and underlying cryptographic primitives. In particular, the use of a secret key enables the generation of a Message Authentication Code (MAC) that will constitute the frame's Security Trailer to provide proof of the integrity of the Frame Data content. Since the key is used to generate the MAC, and it is assumed to be known only to authorized entities, the Security Trailer also guarantees the authenticity of the Frame Data content. The underlying cryptographic primitives of the authentication techniques are, respectively, the Secure Hash Algorithm 2 (SHA2) [10], the Advanced Encryption Standard (AES) [11], and a 128-bit multiplication function over a Galois field $GF(2^{128})$. The CCSDS 352.0-B standard also approves the usage of the Galois Counter Mode (GCM) [9] that combines the CounTeR (CTR) mode [12] of the AES algorithm and the GMAC algorithm to provide authenticated encryption with associated data, i.e. the simultaneous guarantee of confidentiality, integrity, and authentication. In particular, the GMAC algorithm is an authentication-only version of the GCM algorithm, whereas the latter also provides confidentiality by encrypting the Frame Data content. All the indicated algorithms that have been approved by the CCSDS are standardized by the National Institute of Standards and Technology (NIST).

We already presented some preliminary results on a hardware implementation of the authenticated encryption techniques (i.e. the AES-GCM) in [15]. However, we continued and deepened our research on hardware modules for authenticated encryption, developing an improved and configurable architecture that allows for higher frequency and more efficient modules in terms of throughput per resource utilization, which we present in this work.

III. Outline of Authenticated Encryption Modules for Space Security

The AES-GCM algorithm consists of two main functions: the GCTR, for the data encryption/decryption, and the GHASH, for the MAC generation/verification. The GCTR function corresponds to the CTR mode of AES. By denoting the encryption function of AES with $E_{AES}()$, and the encryption and decryption functions of CTR with $E_{CTR}()$ and $D_{CTR}()$, respectively, the encryption and decryption processes of GCTR can be respectively expressed as:

$$C_i = E_{CTR}(P_i, CB_i, K) = P_i \oplus E_{AES}(CB_i, K), \quad (1)$$

$$P_i = D_{CTR}(C_i, CB_i, K) = C_i \oplus E_{AES}(CB_i, K). \quad (2)$$

In (1) and (2), P_i, C_i, CB_i, and K represent, respectively, the ith Plaintext block, the corresponding ith Ciphertext block, the ith Counter Block, and the secret key. P_i and C_i are derived by splitting the input Plaintext (P), or the input Ciphertext (C), in blocks of 128 bits. In the case the last P_i, or C_i, is not a full 128-bit block, it does not require to be padded, since the corresponding C_i, or P_i, is computed by XORing the output of $E_{AES}()$. In such an eventuality, they are used only the most significant bits of $E_{AES}(CB_i, K)$ that corresponds to the bit length of the last P_i (or C_i) block. Instead, the CB_i blocks are 128-bit blocks generated by incrementing by 1 the initial Counter Block (CB_0). This last one is derived from an Initialization Vector (IV) according to its length in bits. In case the bit length of IV is 96, CB_0 corresponds to the concatenation of IV and the 32-bit binary vector 00000000000000000000000000000010. Otherwise, the calculation of CB_0 is more expensive in terms of both resources and time. For this reason, the usage of 96-bit IVs is the most common solution, which is adopted also by the CSSDS [5]. The secret key K can have a bit length of 128, 192, or 256 bits, and the CCSDS approved the usage of only 256-bit keys [5].

The GHASH function relies on the iterative multiplication over a Galois field $GF(2^{128})$ between a 128-bit hash key (H) and a 128-bit data block. At each iteration, the input data block is XORed with the previous product and multiplied by H, as shown by (3).

$$Y_i = (Y_{i-1} \oplus X_i) \cdot H \quad (3)$$

In (3), Y_i and X_i are, respectively, the 128-bit intermediate output block and the 128-bit input data block. Assuming $len(X)$ is the bit length of the input data for GHASH, the number of input data blocks is $m = \lceil \frac{len(X)}{128} \rceil$, and the input sequence is formed by the Associated data (A), the ciphertext (C), and the concatenation of $[len(A)]_{64}$ and $[len(C)]_{64}$. The Associated data are auxiliary data that are not encrypted (using the GCTR function) and are left unchanged in the Frame Data field, whereas $[len(A)]_{64}$ and $[len(C)]_{64}$ are the representation over 64 bits of the bit length of A and the bit length of C, respectively. Hence, the last input block of GHASH (X_m) is a 128-bit block. Instead, it may happen that splitting A and C in 128-bit blocks the last corresponding slice has a length lower than 128 bits: in such a case, the block must be padded to 128 bits by filling it with 0s. According to (4), the hash key H is derived from the secret key K using the encryption function of AES.

$$H = E_{AES}(0^{128}, K) \quad (4)$$

In (4), 0^{128} is a 128-bit block made of all 0s. Finally, the MAC is generated by encrypting the last intermediate result from (3), Y_m, through the GCTR encryption process in (1) and using J_0 as counter block. Therefore, the MAC can be expressed as:

$$T = E_{CTR}(Y_m, J_0, K) = Y_m \oplus E_{AES}(J_0, K). \quad (5)$$

In (5), J_0 corresponds to the concatenation of the IV and the 32-bit binary vector 00000000000000000000000000000001.

According to the CCSDS specifications [5], the AES-GCM algorithm can be applied to all the protocols for TC, TM, Advanced Orbiting Systems (AOS), and Unified Space Data Link Protocol (USLP), by using 256-bit keys, 96-bit IVs, and 128-bit MACs for all of them. Since the GCTR and the GHASH functions are independent, we developed the architecture of a configurable AES-GCM module starting from the individual optimization of the computing units. We used the SystemVerilog language, and, concerning the GCTR unit, we started from the optimization of the underlying AES core. Only the AES encryption function is required as indicated by (1) and (2). The AES encryption function is an iterative algorithm that repeats the transformations *SubBytes*, *ShiftRows*, *MixColumns*, and *AddRoundKey* for a certain number of rounds. In the case of 256-bit keys (AES-256), the number of rounds is 14, and the 256-bit key is used to derive 128-bit round keys through a key expansion routine. The 128-bit round keys are XORed with the data from the *MixColumns* transformation through the *AddRoundKey* step. Only in the last round the *MixColumns* step is skipped and the round key is XORed with the data from the *ShiftRows* step. According to the corresponding literature [16], [17], the most diffused solution for hardware AES modules relies on the single-inter-round-pipelined approach. It consists in the implementation of a single round that is recycled through a multiplexer to select the data from the input or the previous round, and a round buffer to store the intermediate result on each clock cycle. We adopted the same approach and, in addition, we calibrated the insertion of the round buffer along the round path to support the possibility of cascading multiple AES stages for maximizing the throughput [16], [18]. Indeed, although the latency (L) of such architecture is 14 clock cycles, the throughput can be expressed as:

$$THR = \frac{128 \cdot f_{clk}}{CPB}. \quad (6)$$

In (6), f_{clk} is the clock frequency, whereas CPB is the Clock Per Block factor that, similarly to the Clock Per Instruction (CPI) figure of microprocessors, indicates the number of clock cycles required to the AES core to perform the encryption of a 128-bit data block when the pipeline (the chain of AES stages) is continuously fed. The case of a single AES stage is a corner

case in which $CPB = L = 14$. Table I shows the CPB values for the corresponding number of stages (N) supported by our module. We selected all and only the numbers of stages that increasingly lower the CPB according to (7).

$$CPB = \left\lceil \frac{14}{N} \right\rceil \tag{7}$$

Thanks to the calibration of the path delays inside the round path, it was possible to design a module whose critical delay path is expected to not be unaffected by the cascading of multiple stages. Moreover, each AES stage is provided with its own independent Key Expansion Unit (KEU) to derive the 128-bit round keys. Since each KEU for AES-256 requires two 128-bit registers and the number of round keys is 14, we expect that for $N = 7$ and $N = 14$, the usage of a unique and globally shared KEU that computes and stores all the round keys in 14 corresponding 128-bit registers may give benefits in terms of resource utilization. Hence, we developed the corresponding SystemVerilog code to select the instance of the KEUs local to each AES stage or the instance of the global KEU shared among all the stages.

Lastly, we concentrated on the implementation of the S-box module which is the core unit of the *SubBytes* transformation and is the most expensive operation of the whole AES encryption algorithm in terms of both resources and propagation delay. Again according to the corresponding literature, the usage of the approach denoted as Composite-Field Arithmetic (CFA) is traditionally preferred to approaches based on Look-Up Tables (LUTs). It is especially true in Application Specific Integrated Circuits (ASICs) because the CFA approach allows to reduce the problem of finding the multiplicative inverse of a byte (the main S-box operation) to the problem of finding the multiplicative inverse of a 4-bit vector or a 2-bit vector, by significantly reducing the logical complexity of the S-box. Anyway, more recent works [19] highlighted that since the introduction of the LUT6 technology in the FPGA devices, i.e. 6-input LUTs, this trend inverted, preferring the LUT-based implementation of the S-box. Since our target was the space-grade KU060 FPGA that features the LUT6 technology, we developed both versions of the S-box to verify that aspect. However, so as not to preclude the possibility of further investigation and development even on different technologies (standard-cell technologies or other FPGA devices), we made available the selection of the S-box implementation through a configuration parameter at the synthesis level.

TABLE I
SUPPORTED STAGE(S) NUMBER CONFIGURATION AND CORRESPONDING CPB FOR AES-256 IN AES CORE.

Stage(s) number (N)	**AES-256** CPB
1	14
2	7
3	5
4	4
5	3
7	2
14	1

Concerning the GHASH unit, it essentially consists in a Galois multiplier from the hardware point of view. According to the properties of the used Galois field, $GF(2^{128})$, also the integration of a reduction unit is necessary to perform the modulo operation on the 255-bit output product and reduce it to a 128-bit vector. Over the years the corresponding literature [20], [21] proposed the adoption of hardware architectures based on the Karatsuba-Ofman Algorithm (KOA) as the most efficient solution. It consists in splitting the input factors into two halves and using them, and their combinations through an XOR operation, to perform 3 different sub-multiplications. Such 3 sub-multiplications have a lower hardware complexity than the original one, therefore the instance of a unique sub-multiplication unit that is buffered through a register to perform the 3 sub-multiplications (in 3 corresponding clock cycles) favors the reduction of resource consumption. A fourth clock cycle is required to perform the modular reduction. The complexity of the sub-multiplication unit can be further reduced by iteratively applying the KOA, but without inserting additional pipeline stages because, in that case, the advantages in terms of maximum supported frequency would be not able to compensate for the increased latency, resulting in an overall lower throughput. The literature suggests also that the resource complexity scales according to the iterative application of the KOA, but only within a certain limit: after a saturation occurs. For this reason, we developed a configurable GHASH unit with the possibility to select the iteration degree of the KOA for the multi-cycle multiplication unit. In addition, since the AES core can have a CPB equal to 2 or 1 (Table I), we developed also a single-cycle version of the multiplication unit to equalize the latencies between the AES core and the GHASH unit. This is not strictly necessary, as multiple multi-cycle multiplication units can be used in parallel [21]. Using 2 (or 4) multi-cycle multiplication units the overall latency of the GHASH module can be reduced, respectively, to 2 (or 1) clock cycles. However, the usage of parallel multi-cycle multiplication units requires the pre-computation of some powers of H, respectively, up to H^2 and up to H^4, and an equal number of 128-bit registers to store them. Although the expected maximum frequency of the single-cycle multiplication units is lower than that of the corresponding multi-cycle counterpart, such a solution could provide significant resource savings.

Finally, we merged the configurable AES core and GHASH unit with additional logic resources to manage the operations and synchronize the data flow from/to the input/output and the sub-modules, and to pad the data blocks. The overall result was the design of a highly configurable AES-GCM module, whose main architecture is shown in Fig. 3. The main configuration parameters concern the implementation approach of the S-box, the number of cascadable stages (N) in the AES core, the implementation approach of the GHASH unit (single-cycle or multi-cycle), the iteration degree of the KOA in the GHASH multiplication unit(s), and the possibility to instantiate or not the logic resources to perform the decryption/verification process of AES-GCM at the receiver side (box MAC comp. in 3). Accordingly, the dashed boxes in Fig. 3 indicate optional

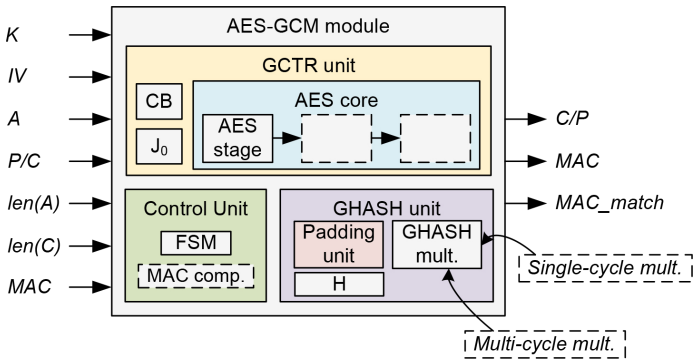

Fig. 3. Outline architecture of the proposed AES-GCM module.

modules. The AES-GCM module was tested through the simulation of a testbench developed in SystemVerilog and including the corresponding official test vectors of the GCM Validation System (GCMVS) [22] released from the NIST through the Cryptographic Algorithm Validation Program (CAVP).

IV. Implementation-aware Trade-offs Analysis

We performed an investigation analysis of the performance of our AES-GCM module based on the implementation on a KU060 FPGA (in particular a device XQRKU060-CNA1509-1M-m) for different combinations of the configuration parameters illustrated in Section III. The corresponding results are reported in Table II and Table III, in which the resource utilization is expressed in Configurable Logic Blocks (CLBs), CLB LUTs, and CLB Registers. A CLB is the main Configurable logic element in the KU060 FPGA (and other FPGAs from Xilinx/AMD), and it includes 8 LUTs and 16 registers. Whether all the LUT and register elements of a CLB are used or only a part is, that CLB is occupied/consumed by the design implemented on the FPGA, so the number of CLBs is the main indicator of resource consumption.

The results in Table II and Table III confirmed all the expectations. Specifically, Table II, through the experiments #1 and #2 (column Run), confirms that on LUT6-FPGA devices the LUT-based implementation of the S-box offers solutions with higher frequency (column Max. freq.) and lower resource consumption (column CLB) than its CFA-based counterpart. The experiments #3 and #4 instead confirm two other aspects. First of all, comparing their maximum frequency against the one of run #2, it can be noted that the cascading approach does not affect the maximum frequency supported by the AES core. Secondly, comparing the CLBs of run #3 against the ones of run #4, it was confirmed that the global KEU allows saving resources for a high number of AES stages (in particular about the 14.2 % for the reported case of $N = 14$).

Table III, on the other hand, gives indications on the GHASH module and its multiplication unit. In general, it can be noted that the single-cycle version of the multiplication unit offers lower frequency (column Max. freq.) and higher resource consumption (column CLB(s)) than the multi-cycle

TABLE II
Implementation results of the AES core on the KU060 FPGA.

Run	N	KEU	S-box version	Max. freq. (MHz)	Resources		
					CLB(s)	CLB LUT(s)	CLB Register(s)
#1	1	Local	CFA	220	343	2093	484
#2	1	Local	LUT	350	267	1733	441
#3	14	Local	LUT	350	2473	13634	5719
#4	14	Global	LUT	350	2121	11387	3964

TABLE III
Implementation results of the GHASH unit on the KU060 FPGA.

GHASH arch.	KOA iter.	Max. freq. (MHz)	Resources			Thr. (Gbps)	Efficiency (Mbps/CLB)
			CLB(s)	CLB LUT(s)	CLB Register(s)		
Single cycle	1	260	776	5574	383	33.28	42.89
	2	250	656	4768	569	32	48.78
	3	230	601	4394	839	29.44	48.99
	4	210	652	4856	1223	26.88	41.23
Multi cycle	1	410	366	2222	879	13.12	35.85
	2	355	295	1892	630	11.36	38.51
	3	350	302	2027	846	11.2	37.09
	4	320	334	2107	862	10.24	30.66

counterpart. In particular, any solution based on the single-cycle approach features a frequency lower than the AES core (with LUT-based S-boxes), hence it can constitute a limiting factor for the whole AES-GCM module. The multi-cycle-based solutions with a number of iterations of KOA equal to 1, 2, and 3 (column KOA iter.) show instead a maximum frequency higher than or equal to the one of the AES core, hence they are the best candidate for the implementation of the AES-GCM module. However, the single-cycle solutions show a higher corresponding throughput (column Thr.) and efficiency in terms of throughput per resource utilization (column Efficiency) because of their reduced latency (1 clock cycle) compared to the multi-cycle solutions latency (4 clock cycles). This result indicates that, according to the requirements of the target application, such as the frequency, the single-cycle solutions may be preferred to multi-cycle solutions in the case a reduced latency is required. For example, in the case of $N = 14$ AES stages and the corresponding $CPB = 1$, the multi-cycle-based approach would require at least 4 parallel GHASH multiplication units, significantly exceeding the resource consumption of the single-cycle counterpart, even if using the multi-cycle solution with the lowest resources cost. Therefore, the usage of only one single-cycle multiplication unit for the GHASH would respect the latency requirement while minimizing the consumption of logic resources in that case. In any case, both the categories of solutions show that 3 is the maximum number of iterations of the KOA that gives benefits in terms of resource consumption; beyond, the resource consumption rises again. This can be appreciated for the single-cycle approach and the multi-cycle approach, respectively, in Fig. 4 and Fig. 5.

A. Comparison with the State-of-the-Art

A fair comparison against other solutions from the state-of-the-art was not quite possible, because the other solutions proposed in the corresponding literature either use different algorithms or are implemented on different FPGAs. In particular, the authors of [23] propose an alternative algorithm

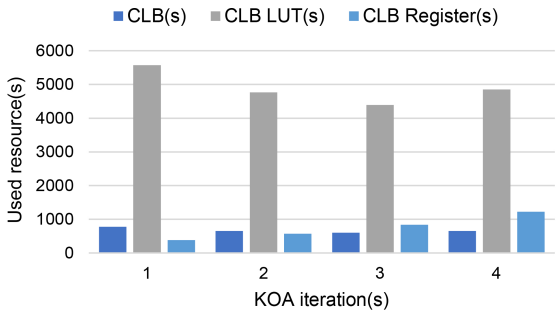

Fig. 4. Resource consumption of single-cycle GHASH.

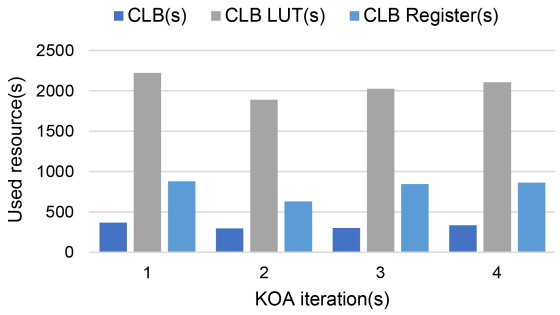

Fig. 5. Resource consumption of multi-cycle GHASH.

to AES-GCM: it is also aimed at authenticated encryption and is based on AES-CTR and IVs, but the function for generating the MAC is very different from GHASH, and the corresponding resource consumption is quite low. In addition, they use a commercial Virtex-6 FPGA, and the adoption of an algorithm that does not perfectly match the AES-GCM specifications makes their solution not fully compliant with the CCSDS specifications. Only the authors of [24] present an implementation of an authenticated cipher module that conforms to the AES-GCM and CCSDS specifications on a space-grade FPGA. However, they used a Virtex-4QV FPGA from Xilinx/AMD (device XQR4VLX200) and did not provide much detail about the architecture of their module. Based on the authors' explanation about the instance of 7 key expansion modules, we configured our module for cascading 7 AES stages accordingly and implemented it on the same FPGA. We also configured our module for the instance of two parallel multi-cycle GHASH multiplication units to support the 7-stage AES core CPB (i.e., 2 clock cycles). The comparison results are reported in Table IV, which shows that our solution supports a higher frequency (Max. freq. column) at a lower resource cost (expressed in slices). This results in a higher supported throughput (Thr. column) and a higher efficiency in terms of throughput per resource utilization (expressed in Mbps/Slice). Finally, the resource usage of our solution also includes dedicated logic resources for GHASH input padding, while the authors of [24] do not explicitly mention this.

TABLE IV
COMPARISON WITH OTHER SOLUTIONS ON A VIRTEX-4QV FPGA.

Work	Padding unit?	Max. freq. (MHz)	Resources (Slices)	Thr. (Gbps)	Efficiency (Mbps/Slice)
[24]	–	139.725	16393	8.94	0.546
This work	Yes	139.8	15530	8.95	0.576

V. USE CASES: SECURE CCSDS-COMPLIANT TRANSMITTERS IN TM APPLICATIONS

As the final step, we implemented two use cases for TM applications by integrating our AES-GCM module in the TM transmitter presented in [25]. Such a transmitter is compliant with the specifications of the standard CCSDS 131.2-B-1 [26] and supports the dynamic change of the corresponding Modulation and Coding (ModCod) schemes. With a maximum frequency of about 120 MHz, the maximum data rate is slightly lower than 2.508 Gbps, and the maximum baud rate is 480 Mbaud. The utilized resources comprise 8887 CLBs and 2.082 Mbit of Block RAMs (BRAMs) on the KU060 FPGA. The power consumption is 2.026 W. The baseline mode for use with TM defined in [5] specifies that the Security Header carries only the 16-bit Security Parameter Index (SPI) and the 96-bit IV (in that order). The SPI is used as an index to retrieve information on the secret key K.

Since the maximum supported frequency of the TM transmitter is 120 MHz, we analyzed two different use cases by integrating the proposed AES-GCM module as a TM Security Module. Its architecture is illustrated in Fig. 6, in which the Input Data interface (I/F) multiplexes and byte-aligns the input for AES-GCM module; the Data Synch Buffer buffers data to be reported unmodified in the SDLS TM Transfer Frame (i.e. the Associated data, A); the Output Data Handler multiplexes and byte-aligns the data formatting the SDLS TM Transfer Frames; lastly, the Parallel-In-Serial-Out (PISO) block, in the Output Data I/F, adapts output data to required parallelism.

A. Single-clock Domain Solution

In this case, we adopted a fully synchronous solution by using the same clock domain (120 MHz) for both the TM Security Module and the TM transmission logic. According to (6) and (7), the minimum number of AES stages required to support the 2.508 Gbps data rate of the transmission module is $N = 3$: indeed, the corresponding CPB is 5 (Table I), and the throughput of the AES-related processes is 3.072 Gbps. Since the CPB is greater than 4 clock cycles, we used the multi-cycle version of the GHASH unit to minimize resource consumption. In particular, we used the one with a KOA iteration degree of 2. We also disabled the instance of the logic resources dedicated to the decryption/verification process that are not required in a transmission. The implementation results are shown in Table V, in which TM transmission and integration logic row includes the data related to the TM transmission logic and the logic required for the integration of the AES-GCM module. The results show that our solution for the authenticated encryption module has a cost of about 3%

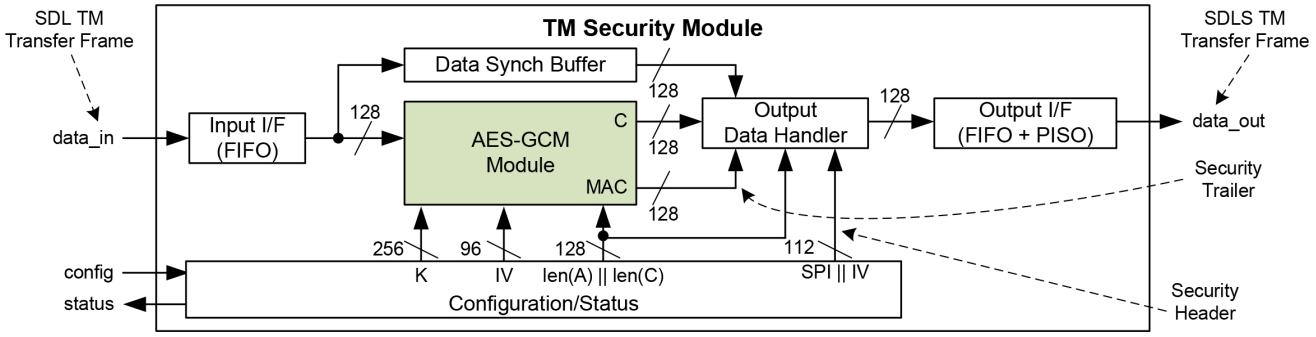

Fig. 6. Outline of the TM Security Module for the two use cases.

of the resources available on the KU060 FPGA and 214 mW in terms of power consumption. They correspond, respectively, to 10.6% and 8.6% of the resource utilization and power consumption of the Secure TM Transmitter.

B. Multi-clock Domain Solution

In this case, we opted for using different clock domains for the TM transmission logic and the TM Security Module to maximize the throughput and minimize resource utilization of our AES-GCM module. According to (6) and (7), our security module requires a minimum clock frequency f_{clk} slightly lower than 275 MHz to support a 2.508 Gbps data rate with a single AES stage ($CPB = 14$). Therefore, we configured the proposed AES-GCM module accordingly and integrated a synchronization interface between the two clock domains. Specifically, we configured our module with a single AES stage, the multi-cycle GHASH unit, and the corresponding multiplier with a KOA degree of 2. Also in this case, we disabled the instance of logic resources dedicated to the decryption/verification process. The implementation results are shown in Table VI, from which it can be seen that the proposed module for authenticated encryption provides the corresponding security services at a cost of about 1.6% of the resources available on the KU060 FPGA and a power consumption of 189 mW. With respect to the entire secure TM transmitter, the resource utilization and power consumption of the proposed solution are 6% and 7.6%, respectively. The amount of resources consumed by the TM transmission and integration logic (TM trasnm. and integration logic) increased with respect to the first use case (Section V-A, Table V) because synchronization mechanisms between the two clock domains were added. However, the total resource consumption was lower than that of the previous solution, as was the power consumption.

VI. Conclusions

In this work, we presented a configurable and scalable architecture for implementing hardware AES-GCM modules aimed at securing space applications compliant with the CCSDS specifications. The proposed architecture was realized as a configurable Intellectual Property (IP) core written in SystemVerilog and compliant with the latest CCSDS security requirements [5], [6]. The results of the implementation on a space-grade KU060 FPGA were (and can be) used as a guide to configure it to find the most efficient solution in terms of resource utilization and throughput per resource, depending on the frequency and/or throughput requirements of the target application. For this reason, we presented two use cases on the integration of the proposed module in a CCSDS-compliant TM transmitter. The corresponding implementation results on the KU060 FPGA showed that our module guarantees comprehensive data security through authenticated encryption with a minimum resource cost of 672 CLBs (1.6%). The usage of 256-bit keys corresponds to a security level of 256 bits in terms of classical security [27], and 128 bits in terms of post-quantum security [28]. Hence our module is able to offer long-term protection (beyond 2031) because the minimum accepted security level is 128 bits [27]. We also performed a comparison with other existing solutions from the literature, showing that our proposal outperforms the competitors. Thanks to the configurable hardware optimization integrated into the SystemVerilog code, it has a higher operating frequency, a lower

TABLE V
Implementation results of the single-clock domain Secure TM Transmitter use case on the KU060 FPGA.

Module	Freq. (MHz)	Resources		Data rate (Gbps)	Power (W)
		CLB(s)	BRAM(s)		
Proposed AES-GCM	120	1235 (3%)	0	3.072	0.214
TM transmission and integration logic	120	10387 (25%)	2.08 Mb (6.4%)	2.508	2.276
Total	–	11622 (28%)	2.08 Mb (6.4%)	–	2.490

TABLE VI
Implementation results of the multi-clock domain Secure TM Transmitter use case on the KU060 FPGA.

Module	Freq. (MHz)	Resources		Data rate (Gbps)	Power (W)
		CLB(s)	BRAM(s)		
Proposed AES-GCM	275	672 (1.6%)	0	2.514	0.189
TM transmission and integration logic	120	10537 (25.4%)	2.08 Mb (6.4%)	2.508	2.286
Total	–	11209 (27%)	2.08 Mb (6.4%)	–	2.475

resource consumption, and a higher throughput, resulting in an overall higher efficiency in terms of throughput per resource utilization. In addition, our solution integrates the padding logic for the GHASH input.

The follow-ups of this work will include the integration of the presented Secure TM Transmitter in a hardware-in-the-loop TM Downlink Emulator [29] consisting of a mass memory emulator, a channel emulator, and the hardware implementations of the transmitter and the receiver for TM protocol. The goal is to evaluate the impact of SDLS services and protocols on TM downlink performances.

ACKNOWLEDGMENT

This work was partially supported by IngeniArs S.r.l. and by the Italian Ministry of Universities and Research (MUR) through the project CN4 - CN00000023 of the program Recovery and Resilience Plan (PNRR), grant agreement no. I53C22000720001, and through the project FoReLab of the program "Departments of Excellence".

REFERENCES

[1] M. Manulis, C. P. Bridges, R. Harrison, V. Sekar, and A. Davis, "Cyber security in new space: analysis of threats, key enabling technologies and challenges," International Journal of Information Security, vol. 20, no. 3, pp. 287–311, 2021.

[2] J. Pavur, and I. Martinovic, "The cyber-ASAT: on the impact of cyber weapons in outer space," in 2019 11th International Conference on Cyber Conflict (CyCon), Tallinn, Estonia, May 28–31, 2019. IEEE: Piscataway, NJ, USA, vol. 900, pp. 1–18, Jul. 2019.

[3] G. Naja, and C. Mathieu, "Space and security in Europe," in Handbook of Space Security, Springer, pp. 371–383, 2015.

[4] The application of security to CCSDS protocols. Informational Report CCSDS 350.0-G-3 (Green Book), CCSDS, Washington, DC, USA, Mar. 2019.

[5] Space data link security protocol. Recommended Standard CCSDS 355.0-B-2 (Blue Book), CCSDS, Washington, DC, USA, Jul. 2022.

[6] CCSDS cryptographic algorithms. Recommended Standard CCSDS 352.0-B-2 (Blue Book), CCSDS, Washington, DC, USA, Aug. 2019.

[7] The keyed-hash message authentication code (HMAC). FIPS 198-1, NIST, Gaithersburg, MD, USA, Jul. 2008.

[8] Recommendation for block cipher modes of operation: the CMAC mode for authentication. SP 800-38B, NIST, Gaithersburg, MD, USA, Oct. 2006.

[9] Recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC. SP 800-38D, NIST, Gaithersburg, MD, USA, Nov. 2007.

[10] Secure hash standard. FIPS 180-4, NIST, Gaithersburg, MD, USA, Aug. 2015.

[11] Advanced Encryption Standard (AES). FIPS 197, NIST, Gaithersburg, MD, USA, Nov. 26, 2001.

[12] Recommendation for block cipher modes of operation: methods and techniques. SP 800-38A, NIST, Gaithersburg, MD, USA, Dec. 2001.

[13] L. Crocetti, F. Falaschi, S. Saponara, and L. Fanucci, "Secure data authentication in space communications by high-efficient AES-CMAC core in space-grade FPGA," in Applications in Electronics Pervading Industry, Environment and Society: ApplePies 2023. Lecture Notes in Electrical Engineering (LNEE); Springer, Cham; vol. 1110, pp. 49–54, Jan. 2024. DOI: 10.1007/978-3-031-48121-5_7

[14] L. Crocetti, F. Falaschi, S. Saponara, and L. Fanucci, "Implementation of a hash-based secure core for integrity and authentication of data in space applications on space-grade FPGAs," 2023 IEEE 2nd Industrial Electronics Society Annual On-Line Conference (ONCON), SC, USA, 2023, pp. 1-6. DOI: 10.1109/ONCON60463.2023.10431035.

[15] L. Crocetti, F. Falaschi, S. Saponara, and L. Fanucci, "Highly-efficient Galois counter mode symmetric encryption Core for the space data link security protocol," in Applications in Electronics Pervading Industry, Environment and Society: ApplePies 2023. Lecture Notes in Electrical Engineering (LNEE); Springer, Cham; vol. 1110, pp. 297–303, Jan. 2024. DOI: 10.1007/978-3-031-48121-5_42.

[16] L. Crocetti, F. Falaschi, S. Saponara, and L. Fanucci, "CRFlex: a flexible and configurable cryptographic hardware accelerator for AES block cipher modes," in Applications in Electronics Pervading Industry, Environment and Society. ApplePies 2021, Online (hybrid), Sep. 21–22, 2021. Lecture Notes in Electrical Engineering (LNEE); Springer, Cham; vol. 866, pp. 117–123, Apr. 2022. DOI: 10.1007/978-3-030-95498-7_5.

[17] R. Ueno et al., "High throughput/gate AES hardware architectures based on datapath compression," IEEE Transactions on Computers, vol. 69, no. 4, pp. 534–548, Apr. 2019.

[18] P. K. Dong, H. K. Nguyen, and X. T. Tran, "A 45nm high-throughput and low latency AES encryption for real-time applications," in 2019 19th International Symposium on Communications and Information Technologies (ISCIT), Ho Chi Minh City, Vietnam, Sep. 25–27, 2019. IEEE: Piscataway, NJ, USA, pp. 196–200, Nov. 2019.

[19] M. Khairallah, A. Chattopadhyay, and T. Peyrin, "Looting the LUTs: FPGA Optimization of AES and AES-like Ciphers for Authenticated Encryption," in 18th International Conference on Cryptology in India: INDOCRYPT 2017, Chennai, India, Dec. 10–13, 2017. Lecture Notes in Computer Science (LNCS); Springer Berlin: Heidelberg, Germany; vol. 10698 pp. 282–301, 2017.

[20] K. M. Abdellatif, R. Chotin-Avot, and H. Mehrez, "FPGA-based high performance AES-GCM using efficient Karatsuba Ofman algorithm," in 10th International Symposium on Reconfigurable Computing: Architectures, Tools, and Applications, ARC 2014, Vilamoura, Portugal, Apr. 14-16, 2014. Lecture Notes in Computer Science (LNCS), Springer Berlin: Heidelberg, Germany; vol. 8405, pp. 13–24, 2014.

[21] K. M. Abdellatif, R. Chotin-Avot, and H. Mehrez, "Efficient parallel-pipelined GHASH for message authentication," in 2012 International Conference on Reconfigurable Computing and FPGAs, Cancun, Mexico, Dec. 5–7, 2012. IEEE: Piscataway, NJ, USA, pp. 1–6, Jan. 2013.

[22] The Galois/counter mode (GCM) and GMAC validation system (GCMVS) with the addition of XPN validation testing. NIST, Gaithersburg, MD, USA, revision Jun. 15, 2016.

[23] S. J. H. Pirzada, A. Murtaza, T. Xu, and L. Jianwei, "Architectural optimization of parallel authenticated encryption algorithm for satellite application," IEEE Access, vol. 8, pp. 48543–48556, Mar. 2020.

[24] D. Muraleedharan, and S. K. Daniel, "An efficient IP core of consultative committee for space data systems (CCSDS) recommended authenticated cryptographic algorithm," in 2020 8th International Symposium on Digital Forensics and Security (ISDFS), Beirut, Lebanon, Jun. 1–2, 2020. IEEE: Piscataway, NJ, USA, pp. 1–6, Jun. 2020.

[25] M. Bertolucci, F. Falaschi, R. Cassettari, D. Davalle, and L. Fanucci, "Comprehensive Trade-off Analysis on the CCSDS 131.2-B-1 Extended ModCod (SCCC-X) Implementation," in 2020 23rd Euromicro Conference on Digital System Design (DSD), Kranj, Slovenia, Aug. 26–28, 2020. IEEE: Piscataway, NJ, USA, pp. 126–132, Oct. 2020.

[26] Flexible advanced coding and modulation scheme for high rate telemetry applications. Recommended Standard CCSDS 131.2-B-1 (Blue Book), CCSDS, Washington, DC, USA, Feb. 2023.

[27] Recommendation for Key Management: Part 1 – General. SP 800-57 Part 1 Rev. 5, NIST, Gaithersburg, MD, USA, May 2020.

[28] V. Mavroeidis, K. Vishi, M. D. Zych, and A. Jøsang, "The impact of quantum computing on present cryptography" International Journal of Advanced Computer Science and Applications, vol. 9, no. 3, 2018.

[29] E. Pagani, R. Ciardi, G. Benelli, M. Bertolucci, F. Falaschi, R. Cassettari, L. Fanucci, and A. Modenini, "CCSDS 131.2-B compliant telemetry downlink hardware-software emulation platform for Next-Generation Earth Observation missions," in 2022 9th International Workshop on Tracking, Telemetry and Command Systems for Space Applications (TTC), Noordwijk, Netherlands, Nov. 28–Dec. 1, 2022. IEEE: Piscataway, NJ, USA, pp. 1–8, Dec. 2022.

[30] L. Baldanzi, L. Crocetti, F. Falaschi, J. Belli, L. Fanucci, and S. Saponara, "Digital Random Number Generator Hardware Accelerator IP-Core for Security Applications," in Applications in Electronics Pervading Industry, Environment and Society: ApplePies 2019, Pisa, Italy, Sep. 11–13, 2019. Lecture Notes in Electrical Engineering (LNEE); Springer, Cham; vol. 627, pp. 117–123, Mar. 2020. DOI: 10.1007/978-3-030-37277-4_14.