Hybrid modular hardware-software solution for securing ground-satellites communication

Enrico Petraglio, Yorick Brunet, Pascal Perrenoud, Clément Dieperink, Yoan Graf,
Anthony Convers, Anthony Jaccard, Yann Thoma
School of Engineering and Management Vaud, HES-SO University of Applied Sciences and Arts Western Switzerland
Yverdon-les-Bains, Switzerland
E-mails: name.surname@heig-vd.ch

***Abstract*—Recent advancements in satellite communication technology have significantly expanded the scope and complexity of space-based applications, from Earth observation to interplanetary missions. As satellites increasingly handle sensitive data and critical infrastructure, ensuring the confidentiality, integrity, and authenticity of transmitted information has become a paramount concern. However, the space environment poses unique security challenges, including high latency, limited computational resources, and vulnerability to cyber threats. To address these challenges, the Space Data Link Security (SDLS) protocol has emerged as a promising standard, offering link security tailored to the constraints of space systems. In response to the growing need for robust security measures in satellite communications, this paper presents a mixed software-hardware modular implementation of the SDLS protocol proposed as a viable solution to improve the performance, resilience and reliability of future space missions.**

***Index Terms*—SDLS, FPGA, Modularity, ASCON, AES-GCM.**

I. Introduction

Secure communication is a critical requirement for modern embedded and aerospace systems, where reliability, performance, integrity, confidentiality and authenticity must be maintained under stringent constraints [1] [2]. The Space Data Link Security (SDLS) protocol, developed by the Consultative Committee for Space Data Systems (CCSDS), addresses these concerns by providing a standardized framework for securing space communication links [3]. While traditionally implemented in software, the increasing complexity and computational demands of secure communication call for more efficient and robust solutions.

In this context, the project described in this paper explores a mixed hardware-software implementation of the SDLS protocol on Field Programmable Gate Arrays (FPGAs), highlighting the key advantages of such an approach. By leveraging the programmable logic (PL) of FPGAs, several critical benefits can be realized. Firstly, FPGA-based implementations allow for parallel processing and pipeline optimization, leading to substantial improvements in cryptographic throughput. These hardware accelerations are particularly advantageous for high data-rate applications, where software-based cryptography might become a performance bottleneck. Secondly, the intrinsic architecture of FPGAs enables secure isolation of cryptographic keys and sensitive data within the programmable logic. This separation from general-purpose processing units adds an extra layer of physical security, making it more difficult for attackers to access or manipulate the cryptographic material [4]. Finally, offloading cryptographic operations from the CPU to dedicated hardware significantly reduces the load on the main processor, allowing it to focus on mission-critical tasks. This not only improves the overall system performance but also enhances real-time responsiveness—an essential attribute in space applications.

This project is funded by Innosuisse, under grant 108.459 IP-ICT.

This paper presents a highly modular FPGA-based implementation of the SDLS protocol, with a design tailored for flexibility and adaptability across diverse mission requirements. The proposed architecture emphasizes modularity at its core, enabling seamless customization and integration. In addition to its structural versatility, the design demonstrates significant advantages in performance and security through comprehensive implementation and evaluation.

One of the primary features of the design presented in this paper is modularity. For this reason, its architecture was designed to be able to accommodate a wide choice of cryptographic algorithms. This configurability allows system designers to select the most appropriate *Authenticated Encryption with Associated Data* (AEAD) implementation depending on mission-specific requirements such as power consumption, data throughput, security level, or resilience to implementation attacks. For a first implementation, two of the most relevant ones were chosen and implemented : AES-GCM and ASCON algorithms.

II. Cryptographic Support: AES-GCM and ASCON

AES-GCM is the de facto cryptographic AEAD algorithm used in the vast majority of space application, including the SDLS standard. Due to its strong security guarantees, high efficiency, and widespread standardization, AES-GCM has been extensively adopted in both terrestrial and space systems for protecting the confidentiality, integrity and authenticity of mission-critical data. However, emerging constraints in modern space missions—including lower-power platforms, increased demand for lightweight cryptographic solutions, and the need for resistance against side-channel attacks—have sparked interest in alternative algorithms. One such algorithm is ASCON, a family of lightweight authenticated encryption

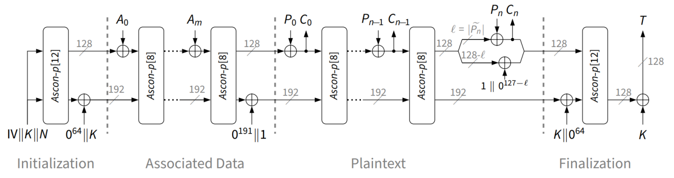

Fig. 1. Internal architecture of ASCON AEAD function [6].

algorithms that won the NIST Lightweight Cryptography competition in 2023 [5]. While AES-GCM is a de facto standard for the SDLS protocol and thus remains essential for compatibility with it, the inclusion of ASCON support introduces an additional layer of adaptability to the platform. The key advantages and characteristics of ASCON, which make it particularly attractive for certain satellite applications are its performance and its small footprint in an FPGA. The NIST competition *Lightweight Cryptography* had the main goal of designing an AEAD algorithm targeted for embedded devices with constrained resources. The central function of ASCON is its permutation function that works on a 320b state. This permutation was designed to be efficient and to hold in a minimum area of an FPGA. The rest of ASCON is similar to a sponge construction [8], which is a small part compared to the permutation function. The ASCON internal architecture is depicted in Figure 1.

III. Design Modularity and Implementation Variants

One of the core strengths of the proposed hardware design lies in its modular architecture, which allows it to be adapted to a wide range of system requirements and mission scenarios. This flexibility makes the design suitable for various levels of integration within FPGA-based platforms, offering scalability in both functionality and resource usage.

The modularity of the system enables the implementation of three primary variants, each tailored to different performance, security, and complexity trade-offs:

Minimal Variant – Cryptographic Coprocessor Only In its most compact form, the design acts as a cryptographic coprocessor, offloading computationally intensive encryption and decryption tasks from the main processing unit. This lightweight variant is particularly well-suited for systems with stringent resource constraints or those that require cryptographic acceleration without full protocol stack implementation in hardware. To enhance flexibility and applicability across a range of use cases, the current coprocessor supports two integrated cryptographic algorithms: AES-GCM [1] and ASCON [2]. They can be selected independently based on the specific security and performance requirements of the target application. Both algorithms were implemented using publicly available reference code and integrated into the hardware design presented and evaluated in this work. It should be noted that both algorithms were chosen from well-documented public repositories. For this work, the effectiveness of the implementation in terms of throughput or resource utilization was not taken into consideration. For this reason, the results reported in this document do not claim to compete with the best implementations on the market. The data obtained during testing serves as a starting point to demonstrate the feasibility of the work presented in this paper.

Intermediate Variant – Cryptography with Key Management A more advanced configuration adds key management capabilities, which handle secure storage, retrieval (only available to the cryptographic coprocessor), and protection of encryption keys. This module not only enhances data integrity but also mitigates threats against unauthorized key access and corruption, improving the system's overall security posture. The keymanager module added in this design variant stores encryption keys either in BRAMS (weaker solution) or for better security into an external non-volatile memory that is exclusively accessible by the FPGA, thereby ensuring strong hardware-level isolation. The processor, which executes the SDLS protocol stack, interacts with these keys solely through an identifier (ID) assigned to each one. Upon receiving an ID, the keymanager locates the corresponding key and verifies its integrity using error correction mechanisms implemented directly within the FPGA design. This architecture guarantees both the isolation of cryptographic keys from the processor and the continuous validation of their integrity, thereby reinforcing the overall security of the system.

Full SDLS Hardware Implementation The most comprehensive implementation embeds the entire SDLS protocol handling within the programmable logic of the FPGA. In this configuration, the hardware autonomously manages SDLS packets formatting, cryptographic operations, and security policy enforcement, while the CPU is reserved for high-level application logic. This variant offers maximum performance and provides the same security isolation as the one presented in the intermediate variant. This full solution is especially suited for systems with stringent real-time or reliability requirements. This implementation option is especially useful if the data to be transmitted is accessible directly from within the FPGA. By eliminating the bottleneck of data transfer from an external memory to the FPGA crypto-core, extremely interesting throughput rates can be achieved.

An overview of the complete hardware implementation, including functional blocks are depicted in Figure 2.

To ease the integration into an existing design, the architecture presented in Figure 2 largely relies on AXI interfaces. All control and configuration operations of the SDLS block are performed via an easy-to-use 32-bit AXI-Lite bus. For optimal performance the data transfer parts, to and from the FPGA, rely on an AXI-Stream bus also of a 32-bit width. The latter allows data bursts of significant size and adds only very little overhead.

To enhance the robustness of the proposed design and ensure its suitability for a broad range of applications in the NewSpace sector, a comprehensive analysis of the architecture

[1]https://github.com/BLu85/AES-GCM-128-192-256-bits
[2]https://github.com/ascon/ascon-hardware

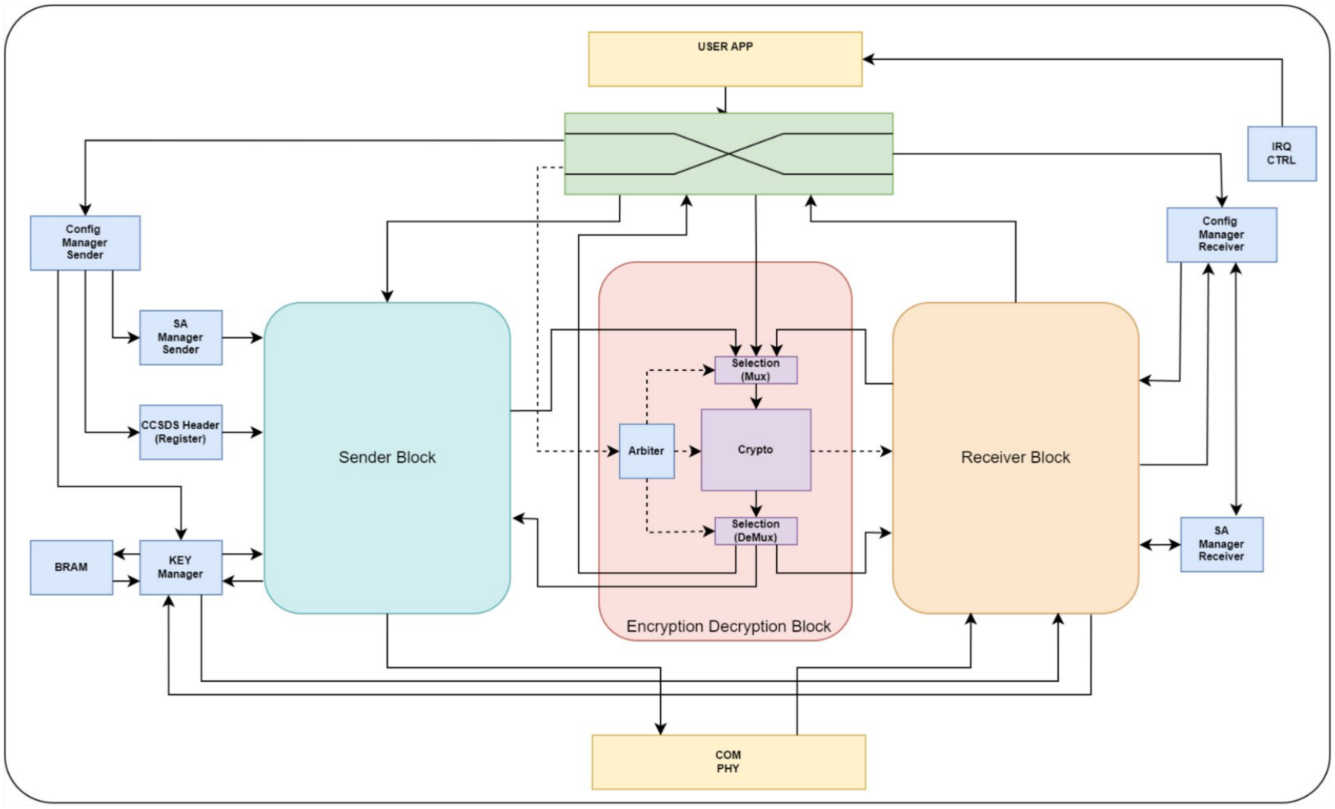

Fig. 2. Full SDLS Hardware Implementation architecture.

functionalities and potential failure points is currently underway. This investigation aims at identifying the most critical and vulnerability-prone components within the system. Based on the findings, targeted fault-tolerance strategies, such as triple modular redundancy (TMR) and the implementation of error correction codes (ECC) [11], will be applied to mitigate risks associated with radiation effects and other environmental hazards typical of space applications. Furthermore, these fault-mitigation techniques will be integrated in a modular fashion, providing end-users with the flexibility to tailor the level of protection in the FPGA implementation according to their specific reliability and resource constraints.

IV. Test Framework

To validate the proposed hardware design, a custom test framework as depicted in Figure 3 was developed, enabling seamless integration between software-based components and hardware-level simulation. The test system leverages existing ground and space components, originally developed for protocol validation and mission planning, and interfaces it with the HDL design under test using the cocotb framework [7]. This hybrid approach allows high-level protocol behavior to be tested directly against the low-level hardware implementation, ensuring functional correctness and system compatibility across abstraction layers.

The initial purely software test framework (blue parts) is composed of: (1) a ground software that is part of the ground infrastructure and implements the SDLS protocol; (2) a space software that is part of the satellite infrastructure and implements the flight control, SDLS protocol, and AES-GCM software crypto backend; (3) an end-to-end test manager that drives the ground software to execute tests. The test framework has been enhanced with green parts: (1) the hardware design implementing AES-GCM and ASCON within the FPGA fabric; (2) the ASCON software crypto backend; (3) the hardware crypto backend that leverages the crypto capabilities of the HDL design; (4) the interface between the hardware crypto backend together with the HDL design under test using the cocotb framework. A mutually exclusive compilation flag instructs the hardware crypto backend which hardware target to address: either the FPGA or the HDL simulation. The test framework runs either fully on a regular computer or with the satellite infrastructure components programmed in a board providing a Processing System (PS) and a PL.

The communication between the hardware backend of the space software and the HDL simulation on cocotb framework leverages eRPC (Embedded RPC) [9], ”an open source Remote Procedure Call (RPC) system for multichip embedded systems and heterogeneous multicore SoCs” as described on its website. The definitions of data types and remote interfaces are written in a generic way following a similar approach as the Protocol Buffers [10]. These definitions are generated in C for the space software and in Python for the test framework, allowing both worlds to exchange data that normally flows through the AXI network or IRQ for tasks synchronization.

The eRPC server functionality is started as a coroutine in cocotb, which runs the generated eRPC server and its locally defined handler. The latter implements remote calls such as *register read*, *register write*, *fifo read*, and *fifo write*. The management of IRQs is being implemented and will run in the reverse direction, the server being implemented on the space side.

The development of the test framework will follow additions to the HDL design, so that new functionalities can be tested right away.

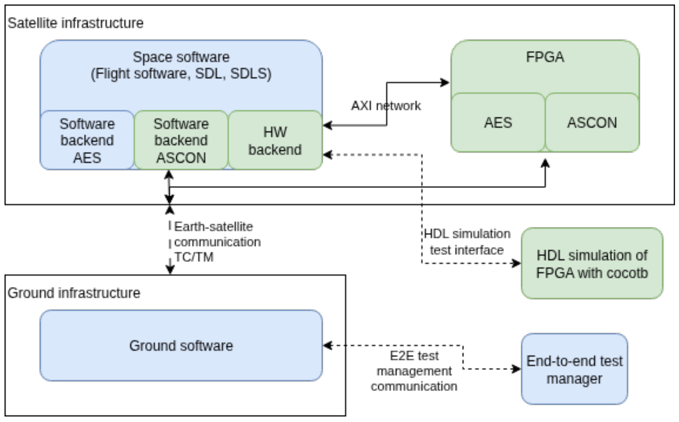

Fig. 3. Test framework (blue: purely software parts, green: hardware-software parts)

V. Results

An initial FPGA-based implementation of the proposed system, incorporating both AES-GCM and ASCON encryption algorithms, was developed and evaluated on the Xilinx ZC706 development board. This implementation represents the minimal configuration of the architecture described in section III and enables the FPGA to act as a cryptographic coprocessor.

To conduct the evaluation, two separate hardware designs were developed: one implementing ASCON and the other implementing the AES-GCM protocol. The resource utilization corresponding to each bitstream is depicted in Table I. The implementation of ASCON demonstrates a substantial reduction in FPGA resource usage compared to the AES-GCM one, with savings ranging from approximately 85% to 93% depending on the resource type. This includes an 84.7% reduction in slice LUTs and logic and 93.4% in slice registers. Such a reduction highlights the significantly lighter footprint of ASCON.

Performance benchmarking was conducted to evaluate the throughput of each architecture. The test involved encrypting and decrypting a message composed of a fixed 128-bit associated data (AD) segment and a variable payload, ranging from 16 to 32,768 bytes. Four implementation variants were tested:

- **AES_soft** and **ASCON_soft**: encryption and decryption are performed entirely in the Processing System (PS) of the FPGA.
- **AES_hard** and **ASCON_hard**: encryption and decryption are offloaded to the Programmable Logic (PL), utilizing the FPGA as a hardware cryptographic coprocessor.

The data rates achieved by all four configurations across varying payload sizes are summarized in Table II, with graphical representations of encryption and decryption performance shown in Figures 4 and 5, respectively. The AES-GCM implementation selected for these tests achieves a peak throughput of approximately 140 MB/s, whereas the ASCON implementation reaches up to 110 MB/s. These performances are strongly influenced by the underlying VHDL design of each algorithm and may be further improved through targeted optimizations and architectural refinements. It is also important to note that the results reported for AES_hard and ASCON_hard account not only for the execution time required to perform encryption and decryption operations, but also include the time needed to transfer data to the FPGA and retrieve the processed output. This approach aims at providing performance measurements that more accurately reflect real-world usage scenarios.

The data transfer operations are implemented using a 32-bit AXI-Stream interface operating at 100 MHz, which yields a theoretical maximum data transfer bandwidth of 400 MB/s. Although the work presented in this paper is still ongoing and further improvements can be made—particularly with regard to the cryptographic implementation, the preliminary results already demonstrate the potential benefits of using an FPGA as a cryptographic coprocessor. Notably, performance gains are observable even for relatively small packet sizes.

When comparing the performance of ASCON and AES-GCM, it is essential to consider not only throughput but also the resource utilization on the FPGA. Although the AES-GCM implementation developed in this work achieves approximately 25% higher throughput compared to ASCON, it does so at the cost of significantly higher FPGA resource utilization. In contrast, ASCON demonstrates a markedly more efficient hardware footprint, making it particularly advantageous in resource-constrained environments where spatial efficiency is critical. To enable accurate benchmarking and ensure sufficient headroom, the designs were initially implemented on the ZC706 development platform, which features a high-capacity FPGA. However, when porting the same designs to a more constrained and widely-used platform—such as the Zynq-7000 (Z-7010) FPGA found on the Zybo development board— the difference becomes stark:

- The AES-GCM core alone consumes over 80% of the total available FPGA resources, exceeding the slice availability and making it impractical without significant performances downgrade.
- In contrast, the ASCON core requires only around 15% of the available resources, offering a lightweight and deployable solution for low-end or embedded platforms.

These results underscore the importance of considering not only throughput but also area efficiency when selecting cryptographic primitives for FPGA-based systems, particularly in applications targeting low-cost or size-constrained hardware.

TABLE I
RESOURCE UTILIZATION COMPARISON BETWEEN AES-GCM AND ASCON CRYPTOGRAPHIC CORES

Name	**Slice LUTs**	**Slice Registers**	**Slice**	**LUT as Logic**	**LUT as Memory**	**Block RAM Tile**
AES-GCM	24553	8331	7093	24553	0	0
ASCON	3723	584	1054	3723	0	0

TABLE II
MEASURED THROUGHPUT FOR THE FOUR EXECUTION VARIANTS IMPLEMENTED ON A XILINX ZC706 WITH A SYSTEM CLOCK @100MHZ

Throughput in [MB/s]			Data payload size [B]											
Algo	**Operation**	**Soft/hard**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**	**2048**	**4096**	**8192**	**16384**	**32768**
AES-GCM	encryption	hard	3.46	5.14	8.55	15.19	25.66	44.19	68.73	96.68	115.52	126.84	132.40	136.27
		soft	2.02	2.43	2.92	3.38	3.71	3.92	4.04	4.11	4.14	4.16	4.16	4.16
	decryption	hard	3.24	4.82	7.98	14.17	24.01	41.65	65.79	94.00	113.34	125.42	131.48	135.85
		soft	1.97	2.39	2.87	3.33	3.66	3.87	4.00	4.06	4.09	4.11	4.11	4.11
ASCON	encryption	hard	3.70	5.53	9.17	14.79	25.38	42.35	63.53	82.74	96.08	103.18	106.40	108.76
		soft	6.28	7.73	9.51	11.21	12.52	13.38	13.88	14.15	14.29	14.36	14.39	14.18
	decryption	hard	3.23	4.84	8.04	13.31	23.89	38.69	59.19	80.29	93.42	101.75	105.72	108.36
		soft	6.26	7.71	9.51	11.24	12.58	13.47	13.98	14.26	14.40	14.48	14.51	14.32

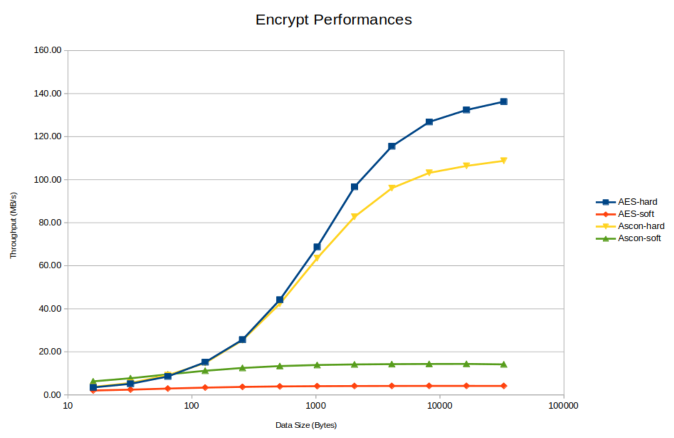

Fig. 4. Graphical throughput representation for encryption.

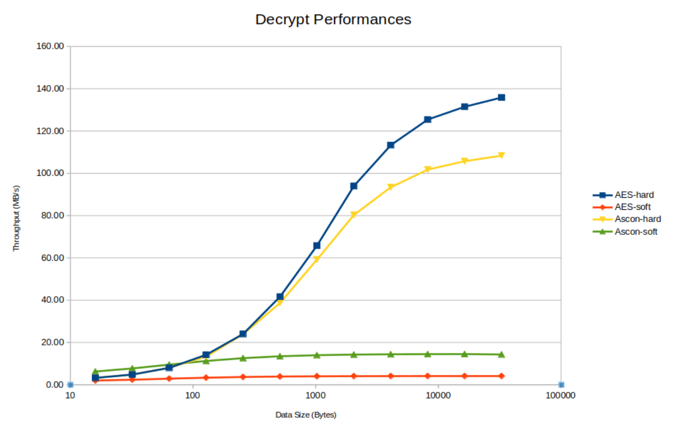

Fig. 5. Graphical throughput representation for decryption.

VI. CONCLUSIONS

This paper presented a fully hardware-based implementation of the Space Data Link Security (SDLS) protocol. Owing to its configurability, the proposed architecture can be tailored to the specific requirements of a wide range of mission scenarios. As a first step toward demonstrating the potential of this design, we evaluated the performance of the FPGA as a cryptographic coprocessor for both the AES-GCM and ASCON algorithms. The results highlight significant improvements, with an observed speed-up of approximately $\times 8$ for ASCON and up to $\times 34$ for AES-GCM, compared to software execution on an ARM Cortex-A9 processor.

The complete system architecture, encompassing all intended functionalities, is currently undergoing simulation and has not yet been synthesized or deployed on an FPGA using the framework described in this paper. Nonetheless, the preliminary results obtained from the cryptographic coprocessor and the key isolation mechanisms that will be provided by the key manager indicate clear advantages, particularly for space-grade applications that possess the necessary hardware resources to support these architectural enhancements.

Furthermore, the full integration of the SDLS protocol, currently under verification and validation, is expected to significantly improve the efficiency of data handling within the hardware architecture. This will enable the FPGA to autonomously manage secure data transfers, thereby substantially increasing the overall system throughput without compromising either security or reliability. Finally, a comprehensive analysis of the design and its critical components is currently in progress. This step will enable the implementation of fault-tolerant protection mechanisms, such as Error Correction Codes (ECC) and Triple Modular Redundancy (TMR), which are essential to further enhance the system robustness and reliability for space mission applications.

A. Acknowledgments

The authors would like to thank Alexandre Duc and Grégoire Guyot for the fruitful discussions concerning AES-GCM and ASCON. They also want to thank CYSEC and their team, as this project, funded by Innosuisse, is part of a collaboration between the School of Engineering and Management Vaud and CYSEC.

References

[1] M. Kang, S. Park, and Y. Lee, "A Survey on Satellite Communication System Security," *Sensors*, vol. 24, no.9, Art.2897, May 2024.

[2] P. Tedeschi, S. Sciancalepore, and R. Di Pietro, "Satellite-based Communications Security: A Survey of Threats, Solutions, and Research Challenges," *Computer Networks*, vol. 216, Art.109246, Oct. 2022.

[3] Consultative Committee for Space Data Systems (CCSDS), "Space Data Link Security Protocol," *CCSDS 355.0-B-1, Blue Book*, Sept. 2019.

[4] K. Järvinen, "Benefits of FPGAs as implementation platforms for cryptosystems," Xiphera Blog, Oct. 26, 2021.

[5] NIST Lightweight Cryptography Team, "Lightweight Cryptography Standardization Process: NIST Selects Ascon," NIST, Feb.7,2023.

[6] M. Sönmez Turan, K. A. McKay, D. Chang, J. Kang, and J. Kelsey, "Ascon-Based Lightweight Cryptography Standards for Constrained Devices: Authenticated Encryption, Hash, and Extendable Output Functions," *NIST Special Publication 800-232 (Initial Public Draft)*, Nov. 8, 2024.

[7] The FOSSi Foundation, "cocotb", www.cocotb.org, accessed: 07.07.2025

[8] Bertoni, Guido, et al. "On the indifferentiability of the sponge construction." Annual International Conference on the Theory and Applications of Cryptographic Techniques. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[9] NXP, "eRPC", https://github.com/EmbeddedRPC/erpc, accessed: 07.07.2025

[10] Google, "Protocol Buffers", https://protobuf.dev, accessed: 07.07.2025

[11] ECSS Secretariat, "Engineering techniques for radiation effects mitigation in ASICs and FPGAs handbook" (ECSS-E-HB-20-40A), published October 11 2023