Design and Implementation of Authenticated Encryption Co-Processors for Satellite Hardware Security

Mohamed H. Abdulmonem[1,*], Ahmed K. Ismail[1,*], and Hassan Mostafa[1,2]
[1]Nanotechnology and Nanoelectronics Department, University of Science and technology, Zewail City, Giza, Egypt.
[2]Electronics and Communication Department, Faculty of Engineering, Cairo University, Egypt.
* These authors have contributed equally to this work.
{s-mohamead.hosni@zewailcity.edu.eg, s-ahmedkhaled@zewailcity.edu.eg, hmostafa@uwaterloo.ca}

2021 International Conference on Microelectronics (ICM) | 978-1-6654-0839-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICM52667.2021.9664957

***Abstract*— FPGA implementation is attained through either the traditional Register Transfer Level (RTL) flow or High-Level Synthesis (HLS) flow. The Consultative Committee for Space Data Systems (CCSDS) has recommended a standard for security algorithms for space missions. Authenticated encryption, the most important of those algorithms, can be achieved by either cipher-based or hash-based algorithms. In this paper, firstly, a brief explanation of the CCSDS standard authenticated encryption algorithms of both types is provided. Secondly, the algorithms are implemented in both RTL and HLS flows to measure and quantify the gap between the two design flows. Results show that the HLS modules utilize 44% more LUTs and consume an average of 40.8% more power than the RTL ones. In addition, the RTL modules demonstrated 28 times higher throughput than that of the HLS ones. Therefore, it is recommended to use the traditional RTL approach over the HLS one and the cipher-based module over the hash-based one at the expense of longer time-to-market for the RTL design. Additionally, the cipher-based module when compared to the hash-based one has proven higher efficiency utilizing 12% less area, achieving 35% higher throughput, and consuming 17% less energy per bit.**

Keywords—HLS, RTL, CCSDS Algorithms, Satellite Security, AES, GCM, HMAC, SHA-2

I. INTRODUCTION

FPGAs have gained a high market share in the hardware security field because of their high adaptability, low cost, and low time to market. An FPGA design can be achieved through an RTL design by hardware descriptive languages or an HLS design using high-level languages. RTL designs give the developer higher design customization and easier architecture exploration. In contrast, HLS designs, where timing is implicit, have faster turnaround times of design and verification. The basic Vivado HLS design flow is shown in Fig. 1 [1].

Space missions need data-system security to prevent unauthorized access to the spacecraft, ground systems, and the data used for spacecraft command or telemetry. Hence, security has become an increasingly important aspect in the space missions design process. The Consultative Committee for Space Data Systems (CCSDS) is an organization established by its member space agencies with the aim of addressing data systems problems common in space missions and devising technical solutions to them. CCSDS issued a recommended standard for security algorithms to be used by space missions [2].

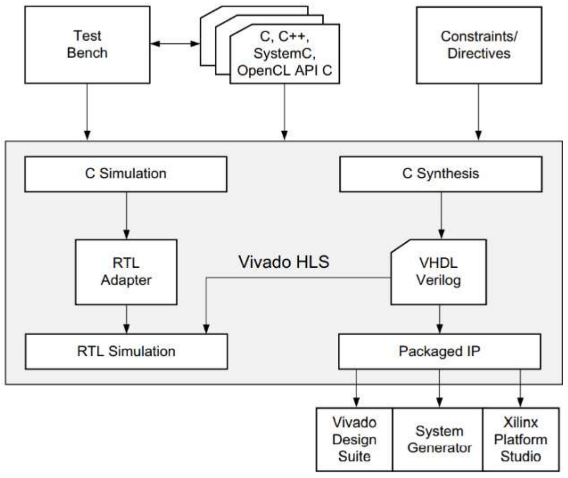

Fig 1. Vivado HLS flow [1].

A. Encryption Algorithms

The CCSDS standard provides recommendations of cryptographic security algorithms for encryption, authentication, and authenticated encryption. Encryption is the only possible method to achieve confidentiality of the information exchanged between the ground and the space. Confidentiality is the assurance that information is not disclosed to unauthorized entities or processes. The encryption algorithm specified by CCSDS is the Advanced Encryption Standard algorithm (AES) [3] using Counter Mode and a 256-bit key.

B. Authentication/Integrity Algorithms

Authentication is the assurance that the claimed identity of the information source is not faked. Data integrity is the assurance that the data itself has not been altered or modified without authorization or notification. Both authentication and data integrity are usually achieved with the same algorithm. CCSDS specifies three alternative algorithms for achieving authentication/integrity. The first one is Galois/Counter Mode (GCM) [4, 5], a cipher-based algorithm. It is simply another mode of operation of the AES algorithm which accomplishes authenticated confidentiality that is the combination of confidentiality and authentication/integrity. The second one is a hash-based algorithm, specifically, Keyed Hash Message Authentication Code (HMAC) [6] using the SHA-256 [7] variant. If HMAC is combined with AES for confidentiality, they can together achieve authenticated confidentiality. The third one is Rivest-Shamir-Adleman (RSA), a digital signature-based algorithm.

C. Satellite Hardware Security

In satellite security, security implementation points include application layer, transport layer, network layer, data link layer, and physical layer [8]. Traditionally, the first 3 layers are achieved through software [9]. For hardware security, the bulk encryption at the physical layer is responsible for the confidentiality of the communication system data structure. At the physical layer, there are no separate services other than encryption that are used for integrity or authentication. However, integrity and authentication are implicitly provided by the encryption key management. On the other hand, the data link layer operates on the data transmission paths, whether it is telecommand or telemetry data. Consequently, a high-security and explicit authentication, integrity and confidentiality security system are required at this layer [9]. Authenticated encryption services can provide these attributes through hardware implementation on FPGAs.

In this paper, RTL and HLS designs of the cipher-based and the hash-based authenticated encryption algorithms are implemented to quantify the gap between RTL and HLS implementations in this application. The digital signature-based authenticated encryption using RSA was excluded as it is usually implemented in software due to the very high hardware area required to implement modular multipliers in a 2048-bit field.

II. CCSDS Algortihms

A. AES Counter Mode

AES is a symmetric-key block-cipher algorithm whose security depends totally on the secrecy of the cryptographic key used. The length of the input block, the output block and the intermediate state block, called block size, is 128 bits divided into four 32-bit words. The length of the key could be 128, 192, or 256 bits [10]. The selected version is the 256-bit key one according to the recommendation of CCSDS. For both its cipher and inverse cipher parts, AES uses a round function composed of four different byte-oriented transformations applied sequentially. The round is executed 14 times in the 256-bit key version of the algorithm. The first transformation is a byte substitution process where each byte is given a new value based on its original value according to a fixed substitution table. Then, the resulting state block is transformed with a process called row shifting where the last three words of the state are cyclically shifted by different offsets. The third transformation is applied on columns of the state. A column is formed by combining all the bytes sharing the same order within their respective words. For example, the second column consists of the second bytes of all the four words. In the third transformation, called column mixing, each column is considered a polynomial in Galois Field (256) and is multiplied modulo x^4 +1 by a certain polynomial. In the last transformation, the state is bitwise XORed with the round key as there is a different key for each of the 14 rounds. Those keys are generated using a specific algorithm from the 256-bit key. As mentioned earlier, the four transformations are executed 14 times. But, there are few exceptions. The column mixing is dismissed in the last round. Also, there is an extra round key XORing before the first round. During the enciphering phase, the plaintext goes through the 14 rounds to produce the ciphertext. During the deciphering phase, the ciphertext undergoes 14 rounds of reverse transformations to obtain the plaintext back. In the counter mode [11], a unique counter undergoes the enciphering phase instead of the plaintext. The resultant block is XORed with the plaintext to form the ciphertext. This way, the same plaintext, if it is sent multiple times, is encrypted with different ciphertexts to ensure stronger security by avoiding pattern repetition.

B. GHASH

GCM algorithm is recommended for authenticated encryption as mentioned earlier. The plaintext is normally encrypted with a ciphertext using the AES counter mode. In addition to the ciphertext, another part of the message is sent without encryption because there is no need to keep it confidential. Then, the ciphertext, its length, the unencrypted data, and its length are all transformed, regardless of their collective size, into a fixed-length block using a hash function called GHASH. It uses a hash key and takes a block with a size that is multiple of 128 bits as input. The first 128-bit block of the input is multiplied by the hash key in Galois Field (2^{128}). Then, the next 128-bit block is XORed with the last multiplication product and multiplied by the hash key and so on until there are no more blocks. The final multiplication product is encrypted using AES counter mode and the resulting ciphertext is considered an authentication tag. The chosen multiplication algorithm consists of 128 identical consecutive steps. Hence, it is possible to implement 128 copies of the hardware of one step and obtain the multiplication result in one clock cycle, implement one copy of the hardware and obtain it in 128 clock cycles, or select any implementation in between. It is worth mentioning that the hash key H is generated by encrypting a 128-bit block of zeroes using AES counter mode. Therefore, the hash key value depends on the secret cryptographic key as well.

C. Cipher-based Authenticated Encryption.

After the authentication tag is generated, the encrypted data, the unencrypted additional data, and the authentication tag are all sent to the receiver. The receiver applies the same hash function with the same hash key to the encrypted and the additional data and encrypts the resulting hash. If the obtained tag is the same as the tag received from the sender, then the data is authentic and will be decrypted. If not, the data is unauthentic, altered or corrupted. So, it shall not be decrypted.

D. SHA 256

SHA-256 is a cryptographic hashing function that digests a message of variable length to 256 bits of data that is unique to the message. Similar to any hashing function, SHA-256 outputs are unpredictable, preimage resistant, second preimage resistant, and collision resistant. SHA-256 uses basic boolean operators and functions. The hash computation starts with message padding that divides the message into 512-bit blocks. The next step is block decomposition where every block constructs 64 words of 32-bit each. The first 16 words are splitting the 512-bit in 32-bit words as in (1) [12].

$$\textit{Message block} = W_1 \parallel W_2 \parallel \cdots \parallel W_{15} \parallel W_{16} \tag{1}$$

Then, the remaining 48 words are constructed as in (2).

$$W_i = \sigma_1(W_{i-2}) + W_{i-7} + \sigma_0(W_{i-15}) + W_{i-16} \tag{2}$$

The hash is computed by executing 64 rounds of the following steps in (3)

$$\begin{aligned} T_1 &= h + \Sigma_1(e) + \mathrm{Ch}(e,f,g) + K_i + W_i \\ T_2 &= \Sigma_0(a) + \mathrm{Maj}(a,b,c) \\ h &= g, \\ g &= f, \\ f &= e, \\ e &= d + T_1, \\ d &= c, \\ c &= b, \\ b &= a, \\ a &= T_1 + T_2 \end{aligned} \tag{3}$$

where "K_i"s are 64 constants given by the fraction part of the cubic root of the first 64 prime numbers. Ch, Maj Σ_0, Σ_1, σ_0, and σ_1 are functions that use basic boolean operators and shifters. Finally, the output is taken as the concatenation of a, b, c, d, e, f, g and h.

E. Hash-based Authenticated Encryption

The basic flow of calculating the HMAC is shown in (4).

$$HMAC\,(K, M) = Hash\,((K \oplus opad) \,\|\, Hash\,((K \oplus ipad) \,\|\, M)) \tag{4}$$

where K is the key and M is the encrypted message using AES in counter-mode. The inner pad (ipad) and the outer pad (opad) make the process of computing the key by trial and error computationally impossible as in order to compute (K ⊕ opad), the (K ⊕ ipad) is needed to the process. This, combined with an AES implementation that drives the input of the HMAC, guarantee an authenticated encryption.

Unlike cipher-based authentication, the authentication code in hash-based authentication is an output of the SHA-256 hashing function. So, the authentication security depends on the complexity of the hashing function. In cipher-based authentication, the authentication code is an output of the ciphering function (AES) where the GHASH digests the message into 128-bit input to the last AES encryption. So, the authentication security depends on the key size of the AES and the key management.

III. Literature Review

A. AES/GCM

Henzen and Fichtner [13] have proposed and implemented a multi-core pipelined FPGA architecture of GCM to achieve a very high throughput of 119 Gbps. However, the resources utilization has been quite high mainly due to using multiple cores. Also, the operating frequency was extremely high which must have manifested in elevated power consumption. Koteshwara et al. [14] have implemented a simpler GCM FPGA architecture which achieved resource usage of 4087 LUTs, 2470 FFs, and power consumption of 19.52 mW. In addition, they have compared the implementation with that of another authenticated encryption algorithm called Deoxys in terms of resources utilization, power, and throughput. Silitonga et al. [15] have developed multiple approaches of different microarchitectures for HLS designs of the AES encryption function only without authentication and compared them with a traditional RTL design on the Zynq-7000 Development Board. They started with a large LUT utilization of 71% in the RTL and 4% in the HLS. They achieved initial throughputs of 889 Mbps and 1.63 Mbps for the RTL and HLS approaches respectively. Then, they improved the microarchitecture used in the HLS approach through pipelining to reach a maximum throughput of 12,800 Mbps and 16% LUT utilization.

B. HMAC/SHA-1,2

In [16], a single-chip cryptographic processor that uses HMAC/SHA-1 has been developed utilizing 310 of 404 IOBs, 7247 CLB slices and 20 Block RAM on a single XCV1000E Xilinx Virtex device. In [17], HMAC was implemented on a FPGA using SHA-1 with the HMAC controller. However, they achieved low throughput considering the high area used. In [18] achieved high throughput of 190 Mbps with energy efficiency 3.2 nJ/b on the Xilinx Virtex-E XCV1600EBG560 FPGA. In [19], a high-speed pipelined design has reached a throughput of 875 Mbps and a total logic cells utilization of 7219 on Altera's Cyclone II FPGA platform. However, the mentioned work only implemented the HMAC algorithm without the encryption function. In hash-based authenticated encryption, AES outputs the ciphertext that is taken as an input to the hashing function.

Nevertheless, no comparison was found in the literature between RTL and HLS authenticated encryption algorithms in terms of FPGA design efficiency. In addition, hash-based and cipher-based authenticated encryption algorithms were not compared in terms of hardware implementation in the literature.

IV. FPGA Implementation

A. Zynq-7000 Development Board FPGA

All implementations were done using a Xilinx Zynq-7000 SoC development board. The board includes a PS of Cortex-A9 processors and a 7-series FPGA. The PL of the Zynq-7020 of our implementations consists of 85,000 cells, 53,200 LUTs, 106,400 FFs, 4.9 Mb BRAM and 220 DSP Slices [20].

B. Cipher-based Authenticated Encryption

As shown in Fig. 2, the top module of the AES/GCM RTL implementation consists of two main datapath blocks. The first main block, named AES Encryption, contains the AES counter mode module connected to the counter incrementing function module. This block is responsible for generating new counters, enciphering the plaintext to obtain the cipher text, enciphering the all-zero 128-bit block to obtain the hash key, and enciphering the hash produced by GHASH to obtain the authentication tag. The same encryption hardware is time-multiplexed to perform the three encryption tasks. The AES counter mode block consists of a generic AES block at the core combined with the required logic for the counter mode operation. The core AES module consists of three smaller modules. The first one is responsible for generating the 14 round keys out of the 256-bit cryptographic key. The second one is a memory-like module used in the byte substitution step in each round. The third one executes the 14 rounds of the AES-256 algorithm. No pipelining of rounds or steps within each round has been implemented. It is the same hardware of one round used sequentially for each of the 14 rounds. The second main block is the GHASH block. It takes the hash key and the ciphertext from the encryption block and returns the hash back to it. It has the Galois Field (2^{128}) multiplication module at the core. As mentioned earlier, that module can be implemented in several ways differing in area and latency. After trying different schemes, it was found that the best compromise was to implement 8 copies of the single step hardware. Hence, the latency becomes 16 cycles. The maximum frequency the RTL reached was 111 MHz. Hence, the respective HLS design was set to be optimized at the same frequency.

C. Hash-based Authenticated Encryption

As shown in Fig. 3, the main blocks needed in an authenticated encryption system using HMAC is AES, SHA-256 and a controller. The AES encryption block is the same one explained previously in AES/GCM. However, the interface of the AES block is 128-bit input/output unlike the hashing function. Consequently, the AES needs to encrypt 4 blocks of 128-bit ciphertext to calculate the input of SHA-256 hashing function for a 512-bit message. Then, SHA-256 calculates the first hash using inner pad and key. The output is used for the second hash using the outer pad and key. Then, the output of the system is the 512-bit encrypted message and the message authentication code from the second hash. The SHA-256 was implemented also to minimize the hardware. So, the 64 rounds are calculated on the same hardware each in a clock cycle. The maximum frequency the RTL reached was 80 MHz. Therefore, the hash-based HLS design was set to be optimized at the same frequency.

V. Results & Discussion

Table I shows the gap between RTL and HLS implementations. Area results are shown for the hash-based and cipher-based modules respectively. Results show 43% increase in LUTs in the hash-based module and 44% increase in the cipher-based module for the HLS design more than the RTL design. Power results have also shown consistent increase in the HLS designs over the RTL designs by 38% for the hash-based module and 43% for the cipher-based module. Also, the cipher-based module, in either RTL or HLS, consumes more power than the hash-based module. The throughput results in Table I show a huge gap between the RTL and the HLS designs. In the hash-based module, the throughput in the RTL design is 27 multiples of the HLS throughout while it is 28 multiples in the case of the cipher-based module. As the cipher-based and the hash-based modules operate on two different frequencies, energy per bit figure of merit was used for a fairer comparison shows the energy per bit results. The cipher-based module is more

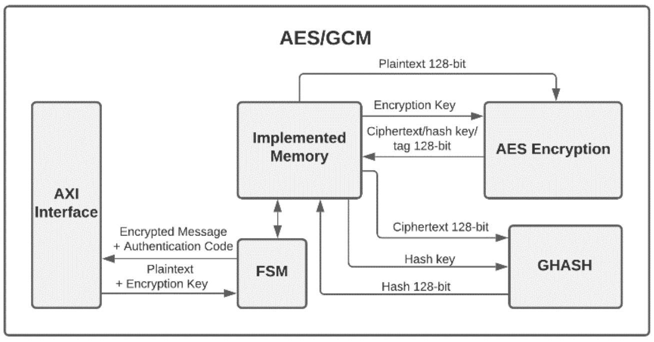

Fig 2. RTL implementation of Cipher-based Authenticated Encryption using AES/GCM.

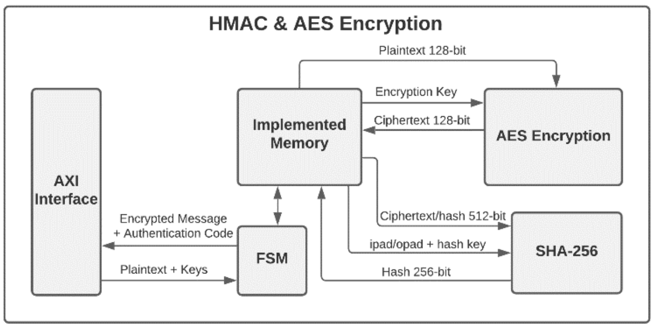

Fig 3. RTL implementation of Hash-based Authenticated Encryption using SHA-256 HMAC and counter mode AES with an incrementing function.

energy efficient than the hash-based module. Also, the hash-based and the cipher-based HLS designs consume 38 times and 40 times more energy than their respective RTL designs.

The hash-based module utilizes more area than the cipher-based module, in either RTL or HLS, as SHA-256 is more complex in implementation than GHASH function. Expectedly, the HLS modules consumes more area than their respective RTL modules. The cipher-based module, in either RTL or HLS, consumes more power than the hash-based module despite it utilizes fewer resources. The reason is that the cipher-based module achieved higher operating frequency of 111 MHz while the hash-based module operating frequency is 80 MHz. Moreover, the HLS designs were set at the same frequencies that their respective RTLs reached for a fairer comparison. Results show higher throughput in the chipper-based module than the hash-based module in both RTL and HLS. That is because the SHA-256 function performs more operations per round than the GHASH function which limits its maximum operating frequency to 80 MHz for minimum hardware.

Table I. The gap between RTL and HLS

	LUTs		**LUT%**		**FFs**		**FF%**		**Power (mW)**		**Throughput (Mbps)**		**Energy (nJ/b)**	
Algorithm	***C***	***H***	***C***	***H***	***C***	***H***	***C***	***H***	***C***	***H***	***C***	***H***	***C***	***H***
RTL	3714	4278	6.98	8.04	4866	5106	4.57	4.8	148	135	154.43	113.15	0.96	1.19
HLS	5352	6111	10.6	11.49	7202	6352	6.77	5.88	213	186	5.53	4.12	38.5	45.15
Gap (difference compared to RTL)	44%	43%	44%	43%	48%	24%	48%	24%	44%	38%	-96%	-96%	3910%	3694%

VI. Design recommendation

Based on the obtained results, the following points should be taken into consideration when designing a hardware-based authenticated encryption algorithm

- The traditional RTL approach is recommended when the time-to-market is not crucial.
- The HLS approach should be avoided if high throughput is required.
- As the gap in the area and power is limited, the HLS is recommended when time-to-market is crucial and no high throughput is required.
- Cipher-based authenticated encryption is recommended in space applications over hash-based authenticated encryption as it has exhibited less resource utilization, more power efficiency and higher throughput than the hash-based authenticated encryption module.
- The hash-based authenticated encryption is recommended when higher security on the authentication is required as the cipher-based authentication hash key depends on cryptographic key used in encryption. On the other hand, the hash-based authentication is achieved by a secure HMAC standalone function independent of the encryption.

VII. Conclusion

Hardware acceleration of authenticated encryption algorithms can be achieved using RTL or HLS flow. The algorithms recommended by the CCSDS were implemented in both flows to quantify the gap between RTL and HLS implementations in this application. Results have shown that the RTL approach more efficient in terms of resource utilization, power, throughput, and energy. It is shown that the HLS modules utilize 44% more LUTs and consume an average of 40.8% more power than the RTL ones. Furthermore, the RTL modules were 28 times as fast as the HLS ones. Besides, the cipher-based module has proven higher efficiency utilizing 12% less area, achieving 35% higher throughput, and consuming 17% less energy per bit than the hash-based one. In the comparison between HLS and RTL approaches, two cases with two different maximum frequencies have been used which strengthen the validity of the results. On the other hand, designing the hash-based and the cipher-based modules for the same maximum frequency would have enhanced the reliability of the results of their comparison against each other. Further optimizations in RTL and HLS modules would increase throughput, area efficiency, and energy efficiency.

VIII. Acknowledgement

This work was partially funded by ONE Lab at Zewail City of Science and Technology and Cairo University, ASRT, NTRA, and ITAC.

References

[1] Introduction to FPGA Design with Vivado High-Level Synthesis UG998, Xilinx, Inc., California, USA, 2013.

[2] *CCSDS Cryptographic Algorithms*. Issue 2. Recommendation for Space Data System Standards (Blue Book), CCSDS 352.0-B-2. Washington, D.C.: CCSDS, August 2019.

[3] FIPS Publication 197, The Advanced Encryption Standard (AES), U.S. DoC/NIST, November 26, 2001.

[4] NIST Special Publication 800-38A, 2001 ED, Version 1, Recommendation for Block Cipher Modes of Operation—Galois/Counter Mode (GCM) and GMAC, December 2001, Natl. Inst. Stand. Technol.

[5] S. Sharaf, and H. Mostafa, "A Study of Authentication Encryption Algorithms (POET, Deoxys, AEZ, MORUS, ACORN, AEGIS, AES-GCM) For Automotive Security", *IEEE International Conference on Microelectronics (ICM 2018)*, Sousse, Tunisia, pp. 315-318, 2018.

[6] FIPS Publication 198, The keyed-hash message authentication code (HMAC), DoC/NIST, March 2002.

[7] NIST, Descriptions of SHA-256, SHA-384, and SHA512, May 2001.

[8] *The Application of Security to CCSDS Protocols*. Issue 3. Report Concerning Space Data System Standards (Green Book), CCSDS 350.0-G-3. Washington, D.C.: CCSDS, March 2019.

[9] M. Manulis, C. Bridges, R. Harrison, V. Sekar and A. Davis, "Cyber security in New Space," *International Journal of Information Security*, pp. 1-25, 2020.

[10] A. M. Ruby, S. M. Soliman, and H. Mostafa, "Dynamically Reconfigurable Resource Efficient AES Implementation for IoT Applications", *IEEE International Symposium on Circuits and Systems (ISCAS 2020)*, Seville, Spain, pp. 1-4, 2020.

[11] Dworkin, Morris. NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation*, National Institute of Standards and Technology, December 2001.

[12] A. H. Gad, S. E. E. Abdalazeem, O. A. Abdelmegid and H. Mostafa, "Low power and area SHA-256 hardware accelerator on Virtex-7 FPGA", *IEEE International Conference on Novel Intelligent and Leading Emerging Sciences (NILES 2020)*, Cairo, Egypt, pp. 1-5, 2020.

[13] L. Henzen and W. Fichtner, "FPGA parallel-pipelined AES-GCM core for 100G Ethernet applications," *2010 Proceedings of ESSCIRC*, pp. 202-205, 2010.

[14] S. Koteshwara, A. Das and K. Parhi, "FPGA implementation and comparison of AES-GCM and Deoxys authenticated encryption schemes," *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1-4, 2017.

[15] A. Silitonga, F. Schade, G. Jiang and J. Becker, "HLS-Based Performance and Resource Optimization of Cryptographic Modules," *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom),* pp. 1009-1016, 2018.

[16] M. McLoone and J. McCanny, "A single-chip IPSec cryptographic processor," *IEEE Workshop on Signal Processing Systems*, pp. 133–138, October 2002.

[17] G. Selimis, N. Sklavos, and O. Koufopavlou, "VLSI implementation of the keyed-hash message authentication code for the wireless application protocol", *10th IEEE International Conference on Electronics, Circuits and Systems (ICECS'03)*, vol. 1, pp. 24–27, December 2003.

[18] Juliato, Marcio, and Catherine Gebotys. "FPGA implementation of an HMAC processor based on the SHA-2 family of hash functions," *University of Waterloo, Tech. Rep,* 2011.

[19] Z. He, L. Wu and X. Zhang, "High-speed Pipeline Design for HMAC of SHA-256 with Masking Scheme," *2018 12th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID),* Xiamen, China, pp. 174-178, 2018.

[20] Zynq-7000 SoC Data Sheet: Overview – Product Specification DS190 (v.1.11.1), Xilinx, Inc. California, USA, 2018.