An Implementation of the 802.1AE MAC Security Standard for In-Car Networks

Berardino Carnevale*, Francesco Falaschi*, Luca Crocetti*, Harman Hunjan†, Samson Bisase†, Luca Fanucci*
*Department of Information Engineering, University of Pisa
Via G. Caruso 16, I-56122 Pisa, Italy
(berardino.carnevale, francesco.falaschi, luca.crocetti)@for.unipi.it, (luca.fanucci)@iet.unipi.it
†Renesas Electronics Europe Ltd
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire SL8 5FH
United Kingdom
(harman.hunjan, samson.bisase)@renesas.com

***Abstract*—The continuous increase in complexity in automotive electronics has led to cars that include up to 80 Electronic Control Units (ECUs). As a consequence, in-car networks are currently up to their limit in terms of data load, flexibility and bandwidth. The Ethernet backbone is thus considered as the best performing solution. On the other hand, the growing interconnection of cars with the external world requires high security standards in order to prevent safety risks for car passengers. The IEEE 802.1AE MAC Security Standard (MACSec) solves the security issues of Ethernet networks by providing confidentiality, authenticity and integrity of data. This paper presents an efficient hardware implementation of the MACSec standard for the automotive world. The system was synthesized on a Stratix V FPGA and on a 28 nm standard-cell CMOS technology. In terms of maximum throughput, the FPGA results in 1.1 Gbps while the standard-cell technology reaches 3.9 Gbps. The FPGA implementation occupies 4.5 % of the Adaptive Logic Modules (ALMs) while the standard-cell one gives a 285 kgate size. The proposed architecture represents a suitable implementation for a low area and high-performance solution as usually required by in-car network controllers.**

I. Introduction

The amount of electronics in the automotive world has dramatically increased due to the flow of new features to improve the driving experience. Cars have become complex systems composed of up to around 80 Electronic Control Units (ECUs) for the highest segment of the market. Therefore, in-car networks are increasingly more complex in order to support communication among this number of devices. Car-to-car and car-to-infrastructure communication also seems to be a reasonable perspective of the Internet of Things (IoT) paradigm in the next few years. The automotive industry should thus offer secure communication in order to protect data flowing over networks, given the increased number of access points offered to a potential attacker. Indeed, automotive network information is often related to or interacts with human safety, and a lack of security can impact on health or lead to a loss of lives. In addition, the increasing interconnection of cars with the external environment could also impact on the transmission of personal data, whose disclosure could have financial or personal consequences.

The most common protocols for in-car networks are: Controller Area Network (CAN), FlexRay, Local Interconnect Network (LIN) and the Media Oriented System Transport (MOST). However, these protocols are currently up to their limits in terms of bandwidth, scalability and connectivity due to the increased complexity of in-car networks and the amount of expected data. In order to solve such constraints, the use of an Ethernet backbone for in-car networks [1], [2] appears to be the next step on which all car manufacturers agree. It would address bandwidth demands and provide reduced complexity by using gateways to connect different protocol sub-domains. Its flexibility could also meet the requirements of car-to-car and car-to-infrastructure communication.

The common idea of adopting an Ethernet backbone should also include the concept of automotive security in order to protect the user against the undesired access of safety- and privacy-related data. The original approach of the 802.3 Ethernet standard [3] does not consider the security problem of data transmitted using such a standard. The Ethernet protocol is known to have several security issues when used in its original form [4]. Using the protocol without any modification therefore does not seem to be the best solution for in-car networks.

In order to solve this problem, the IEEE 802.1AE MAC Security Standard (MACSec) standard [5] was released in 2006. The standard encapsulates the original unprotected frame in a new frame in which data is protected in terms of confidentiality, authenticity and integrity using the 128-bit key Advanced Encryption Standard-Galois Counter Mode (AES-GCM) authenticated-encryption algorithm. The standard also details how frames should be protected against replay attacks but excludes the description of how encryption keys are exchanged among entities. This last concept is covered by the IEEE 802.1X-2010 Port-Based Network Access Control (802.1X-2010) [6] standard. The two following amendments [7], [8] of the standard extended the MACSec to the use of 256-bit keys and improve the replay protection feature.

This paper describes an efficient hardware implementation of the MACSec for the automotive world. The proposed architecture takes into account the limitations and constraints of car industry, by finding a suitable trade-off for such applications. Car ECUs have often space constraints and resource limits that require an area-efficient, low resource-consuming and high-performance network controller. Thus, car industries and consortia [9] consider the 256-bit encryption to be too expensive especially in terms of memory use to store larger

978-1-5090-0366-2/15/$31.00 ©2015 IEEE

size keys. For the previously-mentioned trade-off, our hardware solution only implements the MACSec in its 128-bit key base version without any additional logic to manage improved replay protection.

The paper is organised as follows: Section II briefly illustrates the related work, Section III provides an overview of the MACSec standard, Section IV describes the proposed architecture, Section V analyses the results and finally the conclusions are presented in Section VI.

II. Related work

The need for security in the automotive world has been highlighted by several works that have revealed how low the level of security is in modern cars and how they can be easly hacked [10]. Several European-Union funded projects [11] are also aimed at finding solutions to guarantee security in cars, as done in the past for IT systems. The idea of an Ethernet backbone for in-car networks should take into account the security weaknesses of such protocol highlighted in the literature [12].

The MACSec standard was released to solve these issues in 2006 together with its IEEE 802.1AEbn and IEEE 802.1AEbw amendments in 20011 and 2013 respectively. The standard is based on the AES-GCM algorithm [13] which can be considered as an extension of the Advanced Encryption Standard (AES) [14]. Some AES-GCM implementations [15] are targeted for such standards by focusing on the required Ethernet data throughput.

The MACSec is considered to be the only solution to secure the Ethernet Media Access Control (MAC) layer of the Open System Interconnect Model (OSI Model) stack in network controllers. Several researchers have therefore applied the secure solution to Address Resolution Protocols [16] and Smart-Grid Networks [17]. Various detailed works also focus on Hardware (HW) MACSec implementation, especially for Ethernet Passive Optical Networks [18].

III. The MACSec standard

The MACSec standard provides the methodology to guarantee a secure communication over an Ethernet communication channel by modifying the MAC Layer of the Ethernet OSI Model. The standard is based on the encapsulation of a non-MACSec into a MACSec frame during transmission and vice versa during reception. Entities that want to communicate among them based on the MACSec standard should discover all the nodes connected to the network and create a Security Association (SAs) with them along with a set of Secure Channels (SCs) to communicate. A MACSec frame is composed by the MAC Addresses, the Security TAG (SecTag), The Secure Data, the Integrity Check Value (ICV) and the Frame Check Sequence (FCS) which are obtained as follows:

- MAC Addresses: same as the non-MACsec frame
- SecTag: an improved EtherType bringing all the security information regarding the frame, the network configuration and the replay protection field
- Secure Data: the optionally-encrypted Ethernet Type and Payload of the non-MACSec frame

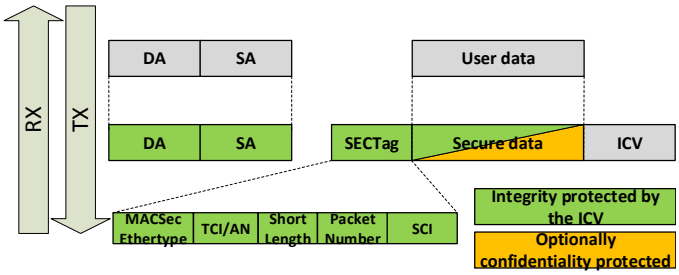

Fig. 1. MACSec frame modifications

- ICV: a 128-bit field whose integrity protects data from the MAC Addresses to the end of the Secure Data

Figure 1 shows the MACSec operations in reception and transmission

Encryption, authenticity and integrity protection of data is performed using the AES-GCM algorithm, whose output is the ICV and Secure Data (User Data if receiving). Therefore, data is protected both from an integrity and (optionally) confidentiality point of view, so that a potential attacker is neither able to understand the frame content without a key, nor able to modify the data or fake its identity without being discovered by the ICV check. The Initialization Vector (IV), which guarantees the strength of the algorithm against brute-force attacks, is derived from the SecTag, thus always guaranteeing a new value for it. Indeed, the SecTag also contains a Packet Number (PN) value, which is unique for every frame. The frame is thus protected against replay attacks and an attacker potentially repeating the same frame is identfied thanks to the unchanged PN.

The set of cryptographic configurations, parameters and algorithms to obtain confidentiality, integrity and authenticity data protection constitute the Cipher Suite. The Cipher Suite takes the key, the SecTag and the data as input, and performs an encryption/decryption and ICV computation/check. Every secure frame communication therefore works if, and only if, the sender and receiver use the same configuration. The generation of SAs, SC and the exchange of keys among nodes are outside the scope of the standard and demanded to the 802.1X-2010.

IV. Proposed solution

In this section the proposed architecture for the MACSec is analysed in detail. In the following subsections the AES-GCM architecture, as well as the MACSec Transmission (TX) and Reception (RX) cores are described, discussing the trade-off and related choices. The implementation can be placed between the Logical Link Control (LLC) and the MAC Layers of the OSI Model Layered architecture. The TX and RX MACSec receive 32-bit words every 4 or 8 clock cycles from the LLC and the MAC Layer respectively. The 4-cycles mode is used for the 1Gbps Ethernet mode while the 8-cycles one is enabled in the 100 Mbps case.

A. The AES-GCM core

The AES-GCM is exactly the same for both the TX and RX core and is compliant with the NIST standard [13].

The algorithm is made up of two main sub-cores: the AES Galois Counter Mode (GCTR) and the Galois Hash Function (GHASH), guaranteeing encryption and integrity protection of data, respectively. The GCTR is based on the AES, the GHASH on a 128-bit Galois Field multiplication. The presented AES-GCM implementation relies on various well-known state-of-art architectural choices.

The main degree of freedom of the GCTR is the S-box implementation of its AES core. From a mathematical point of view, this step computes the multiplicative inverse of each byte of the 128-bit block in GF(2^8). Software implementations often implement the S-box using a Lookup Table (LUT) for each byte, however from a hardware point of view, this would require a large area. The LUT-based approach should thus be avoided given that the final target of the presented architecture is a hardware and area-efficient implementation optimized for automotive ECUs. The S-box step is therefore implemented using a combinational network that presents the best trade-off in terms of area occupation and performance [19].

An additional optimization of the GCTR core is the number of hardware implemented AES rounds required to support the MAC throughput. A round is a single "step" on which the AES algorithm is based, and every n_{th} round receives the 128-bit output of the n_{th-1} one. Below, N_b is the number of bytes per clock cycle received by the AES input buffer, and N_r is the number of AES round,

$$\frac{128}{N_b} > N_r$$

In fact, in our worst-case the input throughput is 1 byte per clock cycle and the AES using 128-bit keys has only 10 rounds. In this case only one round can be implemented in the hardware which can be used iteratively 10 times to perform the algorithm in a folded configuration [20].

Finally the last choice regards the implementation of GHASH's polynomial multiplier. Various optimized solutions are available including the most efficient, which is the Karatsuba multiplier [21]. This reduces the complexity and the size of the multipliers by splitting a large multiplication into several smaller ones. It also entails the intermediate results being aligned and shifted. The Karatsuba multiplier therefore represents the best choice to reduce the size of the system although this is paid for in terms of maximum clock frequency.

B. The TX MACSec core

The TX MACSec module is a system working in the same clock domain of the Physical Layer (PHY) implementation of the Ethernet controller. When enabled, it processes the frames from the LLC and then transfers the generated secure frames to the MAC. It receives information on the kind of frame (i.e. MACSec/non-MACSec) from the LLC, and based on this control signal elaborates or simply forwards the frame itself. Figure 2 illustrates the top level of the TX MACSec module.

The TX MACSec consists of three macro-blocks:

- An input Interface (I/F) to interact with the LLC and accumulate data
- An AES-GCM encryption core to process the data

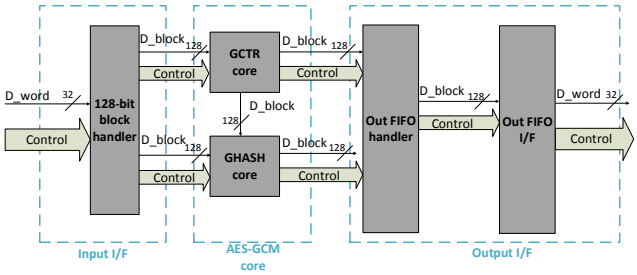

Fig. 2. TX MACSec core block diagram

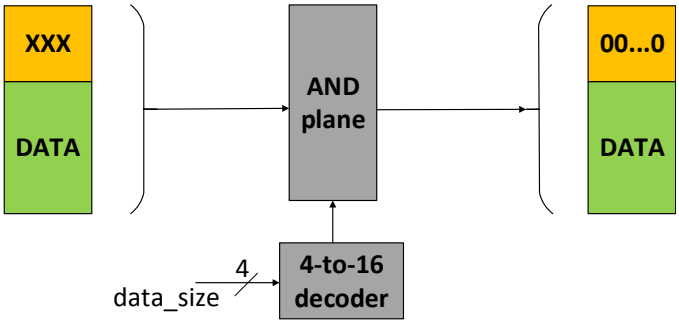

Fig. 3. TX MACSec padding approach, X means don't care

- An output I/F to forward the data after the elaboration

The macro-blocks are made up of several internal blocks, each of which has its own finite state machine. For the sake of clarity, the register interface and the Register Set (RS) are not represented in Figure 2. The RS stores information on the cryptographic keying material, SAs/SC configurations and all the information to be added to the SecTag.

Upon request of the LLC, the Input I/F, begins to accumulate and process data. When it has filled its internal pipeline enough to support the data throughput at the output, it processes the accumulated data through the AES-GCM core and transfers it to the MAC. To read the data from the LLC, the 128-bit block handler of the Input Interface employs a Serial-In-Parallel-Out (SIPO), which receives 32-bit words and produces 128-bit blocks as output, padded as required by the AES-GCM algorithm. An additional feature of the 128-bit block handler module is the combinational network used to pad both the associated data and the ciphertext blocks, as required by the AES-GCM algorithm. This consists of a 4-to-16 decoder, which in accordance with an internal signal appropriate to the size of the current 128-bit block, activates the right number of lines. Each of the sixteen lines is replicated eight times to create a byte mask of all 1s or all 0s depending on the data size. Figure 3 shows the decoding approach.

The output FIFO is a 32-byte register, which is handled by the output I/F. Here a pair of counters are used to highlight the correct location both on the writing side and on the reading side. It therefore works as a Parallel-In-Serial-Out (PISO) buffer receiving 128-bit blocks from the AES-GCM and sending 32-bit words to the PHY implementation of the Ethernet controller.

Finally, the AES-GCM block corresponds to the one described in Section IV-A. It computes the IV depending on the

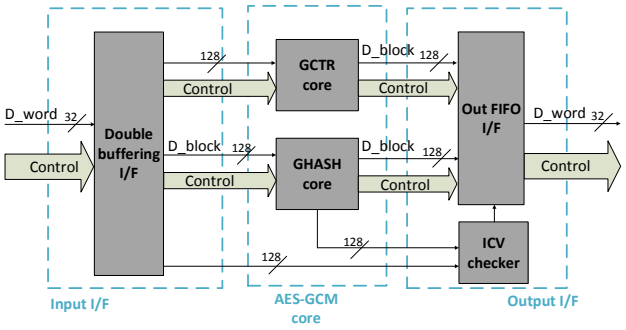

Fig. 4. RX MACSec core block diagram

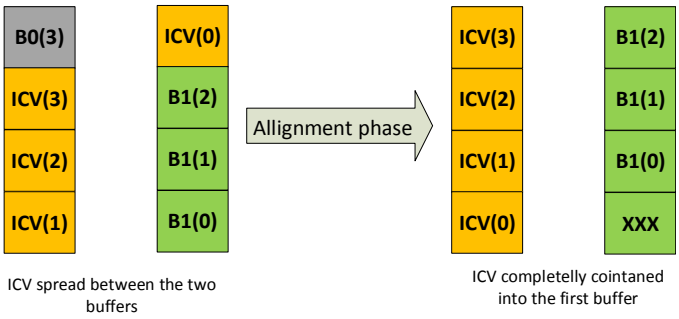

Fig. 5. TX MACSec core ICV alignment. The green section of data are the ones to decrypt using the AES-GCM. The data in yellow belong to the ICV and are aligned in the first buffer using the two buffers as a unique shift register.

RS content, which is also used to prepare the SecTag. The PN of each m_{th} frame corresponds to the m_{th-1} increased by one.

C. The RX MACSec core

The RX MACSec is a system clocked with the clock of the PHY layer of the Ethernet controller. Unlike the TX, it does not need to know from the previous block whether the frame is a MACSec/non-MACSec because it can extract such information from the frame's Ethernet Type, which for a MACSec frame is always 0x88E5. It therefore controls its own the decision regarding the use of the Secure/Unsecure channel.

The RX MACSec consists of three macro-blocks:

- An input I/F to accumulate data
- An AES-GCM encryption core to process the data
- An output I/F to transfer the processed data

Figure 4 shows the top level RX MACSec module. The macro-blocks are composed of several internal blocks, each of which has its own finite state machine. The RX MACSec module conceptually works like the TX MACSec, but with the opposite data flow. For the sake of clarity the register interface and the RS are not represented in Figure 4. The RS stores information about cryptographic keying material and SAs/SC configuration.

The input block is made up of the double buffering interface. This macro-block is also responsible for identifying the kind of frame from its Ethernet Type and sets the MACSec/non-MACSec choice. The input double buffering approach is required to detect the ICV at the end of the received frame and to align it in a single 128-bit buffer, as shown in Figure 5. A 128-bit block can be sent to the GCTR and GHASH core only when it is complete. If the 128-bit block is not complete, it should be padded, as required by the AES-GCM standard, and its ICV should be saved in order to be able to check its integrity. The double buffering allows this behaviour because the second-to-last 128-bit block is sent only if the following 128-bit block does not contain the ICV. Otherwise, it is padded with the appropriate number of zeros and the ICV is aligned in the last buffer using the two buffers as a shift register. On the other hand, the input padder inserts the right number of zeros when a non-complete block needs to be elaborated.

TABLE I. LATENCY OF THE MACSEC IN CLOCK CYCLES

MACSec frame	1 Gbps	100 Mbps
TX: Only integrity	34	67
TX: Integrity and confidentiality	34	67
RX: Only integrity	43	75
RX: Integrity and confidentiality	39	67

The output interface receives 128-bit blocks from the AES-GCM core and, working as a PISO buffer sends the 32-bit words to the LLC. This block also sends information to the output regarding the integrity check and replay attacks. The integrity check is obtained comparing the ICV computed by the GHASH with the received one stored in the first buffer of the double buffering interface. The replay protection is obtained by comparing the expected PN stored in the registers with the one saved in a Packet Numbering buffer block. The results of such checks are passed to the LLC using a STATUS word, which is an additional word sent to the LLC at the end of the data (i.e. after the ICV) storing additional information about the received frame.

Finally, the AES-GCM block is exactly the same as the before described one described in Section IV-A. It receives the Initialization Vector and the data from the input Double Buffering Interface. The timing interaction between the GCTR and GHASH, as required by the AES-GCM is directly managed by the bock Finite State Machine (FSM).

V. RESULTS

In this section, the proposed architecture is analyzed in terms of latency and compared with other state-of-the-art solutions of the MACSec standard.

The required clock frequency of the MACSec is 125 MHz, that is the same clock domain as the MAC in which it was integrated. This MAC supports both the Ethernet 1 Gbps and 100 Mbps modes which are considered sufficient for automotive applications [9]. The latency in terms of clock cycles is given by the initial accumulation phase of a 128-bit block and depends on the speed of the Ethernet. Table I depicts the latency of the two blocks. As shown, the number of latency clock cycles is small and suits the low response time required by the automotive industry.

The throughput of the implementation is not modified by the two blocks and results in 100 Mbps and 1 Gbps for the two

TABLE II. COMPARISON WITH OTHER IMPLEMENTATIONS

Implementation	Gate count (kgates)	Throughput (Gbps)
This work	284	3.9
[18]	350	1

configurations.

The two modules were implemented on the Altera Statix V 5SGXMABK3H40C4, a 0.9 V, 28 nm FPGA and synthesized at the required clock frequency of 125 MHz. The occupation of the TX MACSec and the RX MACSec both including their own AES-GCM core, is 2.11 % and 2.37 % of the ALMs with an overall occupation of 4.48 % of the FPGA. The maximum clock frequency reachable by the same implementation is approximately 148.1 MHz and 149.6 MHz for RX and TX, respectively. This leads to a maximal theoretical throughput of 1.18 Gbps when a single byte is transmitted or received for each clock cycle and both RX MACSec and TX MACSec are implemented in the same Ethernet controller.

In addition, the two systems were synthesized on a 28 nm standard-cell CMOS library. Results show that the complexity of the systems is 156 kgates for RX MACSec, and 128 kgates for the TX MACSec, with an overall occupation of 284 kgates. The maximum clock frequency on the same technology is 486 MHz and 495 MHz for RX and TX, respectively. At this frequency, the maximum achievable throughput is 3.9 Gbps.

Table II compares the standard-cell results with the state-of-the-art. The table highlights that the system presents an optimized implementation for the Ethernet 100 Mbps and 1 Gbps considering that the gate size is 23% smaller, while the throughput is almost 4 times higher. The architecture presented in this paper would thus be suitable for automotive applications in which small circuits with a good performance are required.

VI. CONCLUSIONS

This paper has presented an efficient hardware solution implementing the MACSec standard. This implementation is suitable for automotive applications that will integrate the Ethernet backbone. The system is compliant with the standard but does not implement its amendments due to car network area constraints. When implemented on FPGA and standard-cell technologies, the solution is attractive in terms of area occupation and throughput.

Future work will involve the integration of the architecture with the 802.1X-2010 standard in order to create a stand-alone system capable of also performing channel creation and key exchange.

REFERENCES

[1] T. Steinbach, K. Muller, F. Korf, and R. Rollig, "Demo: Real-time ethernet in-car backbones: First insights into an automotive prototype," in *Vehicular Networking Conference (VNC), 2014 IEEE*, 2014, pp. 133–134.

[2] L. Bello, "Novel trends in automotive networks: A perspective on ethernet and the IEEE audio video bridging," in *Emerging Technology and Factory Automation (ETFA), 2014 IEEE*, 2014, pp. 1–8.

[3] *IEEE Standard for Ethernet*, IEEE Std. 802.3.

[4] W.-j. Sun and H. Cai, "Ethernet switch-based security risks and defense method research," in *Computational and Information Sciences (ICCIS), 2011 International Conference on*, 2011, pp. 765–768.

[5] *IEEE Standard for Local and Metropolitan Area Networks-Media Access Control (MAC) Security*, IEEE Std. 802.1AE-2006, 2006.

[6] *IEEE Standard for Local and metropolitan area networks - Port-Based Network Access Control*, IEEE Std. 802.1X-2010 (Revision of 802.1X-2004), 2010.

[7] *IEEE Standard for Local and metropolitan area networks–Media Access Control (MAC) Security Amendment 1: Galois Counter Mode–Advanced Encryption Standard– 256 (GCM-AES-256) Cipher Suite*, IEEE Std. 802.1AEbn-2011 (Amendment to 802.1AE-2006), 2011.

[8] *IEEE Standard for Local and metropolitan area networks–Media Access Control (MAC) Security Amendment 2: Extended packet numbering*, IEEE Std., 2013.

[9] [Online]. Available: http://www.evita-project.org/

[10] P. Mundhenk, S. Steinhorst, M. Lukasiewycz, S. Fahmy, and S. Chakraborty, "Security analysis of automotive architectures using probabilistic model checking," in *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*, 2015, pp. 1–6.

[11] [Online]. Available: http://www.evita-project.org/

[12] P. Feng, "Wireless LAN security issues and solutions," in *Robotics and Applications (ISRA), 2012 IEEE Symposium on*, 2012, pp. 921–924.

[13] *NIST 800-38D (2005) Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*, Std., May 2005.

[14] *Announcing the Advanced Encryption Standard (AES), FIPS-197*, Std., 2001.

[15] C. Zhang, L. Li, J. Xu, and Z. Wang, "High-throughput GCM VLSI architecture for IEEE 802.1ae applications," in *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, 2009, pp. 900–903.

[16] J.-W. Lee, S.-H. Park, K.-H. Gum, and T.-M. Chung, "Design of secure arp on MACsec(802.1Ae)," in *Ubiquitous Information Technologies and Applications (CUTE), 2010 Proceedings of the 5th International Conference on*, 2010, pp. 1–4.

[17] N. Indukuri, "Layer 2 security for smart grid networks," in *Advanced Networks and Telecommuncations Systems (ANTS), 2012 IEEE International Conference on*, 2012, pp. 99–104.

[18] K.-S. Han, K.-O. Kim, T. W. Yoo, and Y. Kwon, "The design and implementation of MAC security in epon," in *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*, vol. 3, 2006.

[19] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Efficient and high-performance parallel hardware architectures for the AES-GCM," *IEEE Trans. Comput.*, vol. 61, no. 8, pp. 1165–1178, 2012.

[20] R. Chaves, G. Kuzmanov, S. Vassiliadis, and L. Sousa, "Reconfigurable memory based AES co-processor," in *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, 2006.

[21] J. Wang, G. Shou, Y. Hu, and Z. Guo, "High-speed architectures for ghash based on efficient bit-parallel multipliers," in *Wireless Communications, Networking and Information Security (WCNIS), 2010 IEEE International Conference on*, 2010, pp. 582–586.