Efficiency of Cryptography for Multi-Algorithm Computation on Dedicated Structures

Nicolas Sklavos[I], João Carlos Resende[II], Ricardo Chaves[II], Francesco Regazzoni[III], Osnat Keren[IV]

[I] KNOSSOSnet R.G., CIED Dept., Technological Educational Institute of Western Greece, Greece

[II] INESC-ID, IST, University of Lisbon, Portugal

[III] ALaRI Institute, University of Lugano, Switzerland

[IV] Bar Ilan University, Israel

Abstract— **Currently, the more efficient implementations for cryptographic engines are dedicated structures and only allow processing a single algorithm. On the other hand the existing multi-algorithm processors impose significantly higher costs, resulting in low efficiency structures. One possible solution is to use reconfigurable systems. However reconfigurable technologies impose additional costs, both in efficiency, as well as in the reconfiguration process itself. The solution herein proposed considers the support for multi-algorithm by looking into the common operations and merging there computation into common components. The obtained results suggest that by proper component reuse and adequate scheduling, efficiency and cost metrics identical to those of dedicated structures can be achieved, while supporting the computation of multiple algorithms. As future perspective, the implementation of security codes against Differential Fault Analysis, in such multi-algorithm architectures, is studied.**

Keywords—multiple algorithms, FPGA, AES, CLEFIA, security codes, circuits & systems, hardware design, Differential Fault Analysis attacks

I. Introduction

In the near future almost each aspect of our everyday live will be pervaded by a number of devices connected to form the so called Internet of Things (IoT). These devices will include some sort of computational power, sensing capabilities, and of course, network connectivity. Additionally, considering the potential sensitivity of the data involved and the eventual use in critical applications, it is possible to envision that cryptographic functionalists will be included too. However, each device, will be designed to address a specific function, which is characterized by its specific requirements and applications constraints. This will also affect the type of cryptographic primitives which will be included into devices, and the way in which they will be implemented. Devices which have to fulfill extremely strict requirements in terms of power and energy will support only lightweight algorithms, such as PRESENT [1] or KATAN [2]. Others, which have more relaxed constraints, will support more complex ones such as AES [3]. Nevertheless, they have to guarantee and support connectivity, as they will all be connected together. The communication among devices, implementing different algorithms, would require the presence of a node acting as hub. Devices would then communicate with a hub, supporting a number of different algorithms and schemes, which will support different protocols and allow the interoperability.

Although, different cryptographic algorithms have a common or similar structure. Often there is a non linear step implemented using Look-Up-Tables (LUTs), a diffusion layer, and a key addition carried out, using XOR gates. As a result, it would be in principle possible to integrate several algorithms into a single architecture, achieving a minimum cost in terms of area, while maintaining an acceptable level of performance. A simple and cost effective way to implement this functionality exploits the potential of reconfigurable hardware. FPGAs have been used so far mainly as prototyping devices and address market segments, characterized by low volume. But it is possible to envision that their usage will be increased more, and that they will be integrated in future processors.

Towards this, the paper explores a scenario where multiple cryptographic functions co-exist on the hub. In this context, this work proposes a design that supports a range of different block ciphers, realized on top of a common base architecture. It is shown that by a detailed analysis of targeted algorithms, it is possible to efficiently reuse computational blocks, to implement compact multi-algorithm encryption engines. To achieve this, the operations of each algorithm have to be properly scheduled, and some of the components have to be designed to perform different operations, such as the common S-Box operation (in this case T-Box).

If the available computational resources of the supporting technologies are properly explored, it is possible to derive cryptographic engine, supporting multiple algorithms, with high throughput and relatively low area footprints. In this particular case, considering the Xilinx VIRTEX 5 technology, a dual AES/CLEFIA cryptographic engine achieving throughputs between 1 Gbps and 850 Mbps, at a cost of 123 Slices and 3 BRAMs, is achieved in this work.

The last part of this paper, deals with future design perspectives; the implementation of security codes against Differential Fault Analysis (DFA), in multiple algorithms architectures, is considered. Codes that can detect weak attacks, as well as codes for strong ones are introduced.

II. Design Of Reconfigurable Systems

Reconfigurable systems are emerging as key technologies, towards dedicated ones, while still being highly adaptable and with high efficiency computation structures. This technology also allows to be dynamically adapted to the particular computing requirements, thus reducing the overall cost of the system. The last feature is very useful in systems that need to be remotely configured/adapted, or that operate on heterogeneous environments, such as the IoT. Reconfigurable systems use highly flexible computing fabric, usually the Field Programmable Gate Arrays (FPGAs). The partial dynamic reconfiguration functionality of modern FPGAs enables the device to repeatedly change its configuration while operating. In [4], it is shown that the usage FPGAs in the nodes of heterogeneous clusters, can result in significant speedup and energy reductions, in comparison to the traditional clusters.

The main components of these reconfigurable systems are a set of programmable logic blocks, typically implemented as small programmable Look-Up-Tables (LUTs) and interconnection matrixes, composing configurable network of logic operations. The adaptation of the network is performed by a configuration bitstream defining the content of each Look-Up-Tables (LUTs), and thus their logical operation, and their interconnection. This configuration bitstream is topically stored on an external memory and loaded into the reconfiguration device, when it is needed to be reconfigured.

However, this flexibility comes at a cost, particularly in terms of reconfiguration time, energy, and security. The reconfiguration time is imposed by the need to update the internal state of each Look-Up-Table (LUT) and interconnection matrix. This also comes at an energy cost, since this update is also energy consumed. Security wise, the resulting threats are mainly due to the fact that most FPGAs are volatile and the configuration data needs to be stored on, potentially unsecured, external memory and sent into the device. If the configuration data are not authenticated before being loaded, the device itself may be compromised. These threats are particularly relevant when the system is deployed in a hostile or public environment [5]. To mitigate this problem FPGAs provide bitstream encryption and authentication mechanisms. Modern reconfigurable devices include on-chip AES decryption engine, use Cyclic Redundancy Check to validate the integrity of the configuration and perform bitstream authentication using strong hash functions.

However, several attacks can still be performed, such as system downgrade attacks [6] and corruption of reserved regions [7]. Several solutions have been proposed to further minimize these threats [7-9]. However, all these present added costs in terms of area and energy.

III. Multi-Encryption System

Given the cost and possible threats that reconfiguration implies, an alternative solution can be the reuse of computational structures, in order to provide support for multiple algorithms. Towards this, multi-cryptographic co-processors have been proposed [10-12]. These resembled general processors, making them suitable for general encryption systems, with lower encryption throughputs. However, given the general approach they are not able to provide adequate efficiency metrics and also impose significantly higher energy consumption when compared with dedicated structures. In order to provide adaptable encryption structures, but with high efficiency and low footprint, cryptography engines, dedicated multi-encryption structures capable of efficiently computing a restricted set of algorithms can be considered [13].

In order to provide a proof of concept that multi-encryption structures capable of high throughputs with relatively low area footprints can be achieved two very different symmetrical block ciphers are herein examined, AES and CLEFIA.

The well known AES algorithm [3] is a 128-bit block cipher based on a Substitution-Permutation Structure. It accepts 128-, 192-, and 256-bit long keys processed over 10, 12 or 14 rounds, repetitively. After the initial addition with the first round (performed by the XOR operation), the input data goes through several operation on each round. The round computation is composed of *SubBytes*, where each byte is replaced by another one, which can be implemented by a Look-Up-Table (SBox); *ShiftRows*, where the second to fourth row of the State are left-round shifted one to three bytes, respectively; *MixColumns*, where each column of the State is multiplied over GF($2^{\wedge 8}$), where the entire State is XORed with the corresponding 128-bit Round Key. The decryption process of the AES cipher is performed identically to the encryption, but with the inverse operations [14].

The CLEFIA algorithm is a 128-bit block cipher, based on a 4-branched Feistel network, accepting key lengths of 128-, 192- and 256-bit, processed over 18, 22, or 26 rounds. The 128-bit (16 bytes) of plain text are arranged into an array of four 32-bit words. The first step of the encryption process is to XOR the second and fourth words of the input data with the first and second 32-bit of the original key, performing a key whitening procedure. After this operation the rounds are executed. In each round, the four input words are processed, where copies of the first and third input words go through a 32-bit output non-linear function, F0 and F1 respectively. In each one, the result is XORed with the second and fourth words. The resulting four words are then swapped by left-round shifting them [13].

Despite having different computation structures, both share similar ciphering techniques, such as diffusion matrices and byte substitution (SBoxes).

IV. Proposed Multi-Algorithm Architecture

Towards the targeted multi-encryption structure, the state of the art compact structures for dedicated AES [14-18] and CLEFIA [19-21] ciphers are considered. As the prototyping platform, Field Programmable Gate Arrays (FPGAs) is herein considered as targeted technology, given their increasing deployment in embedded systems, adaptability, and ease of prototyping.

The proposed architecture [17] considers the use of a folded round structure, computing each round in multiple iterations with a datapath of 32-bit using a TBox approach, as depicted in Figure 1.

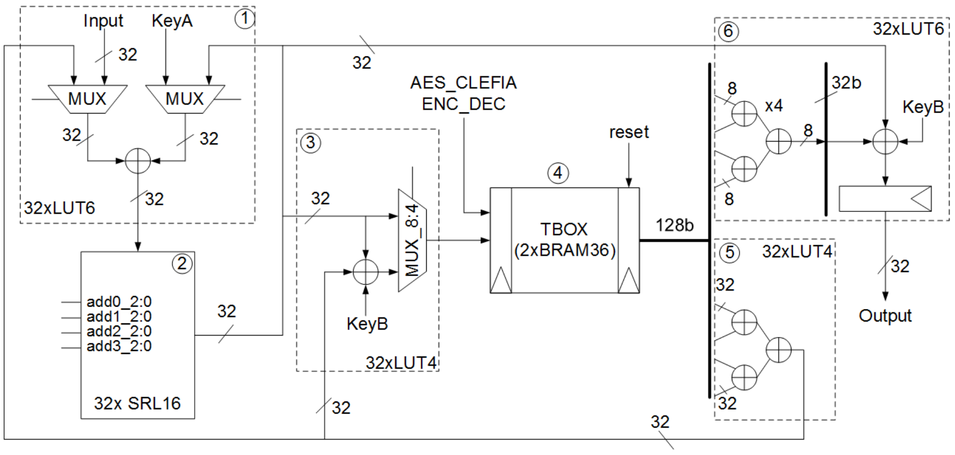

Fig. 1. Proposed AES/CLEFIA Datapath

The 32-bit folded structure is suggested by the state of the art to be the best compromise towards a compact structure still capable of achieving high throughputs.

Since the initial step for both algorithms is to XOR the plaintext with the first input keys, a 32-bit XOR operation is considered, as depicted by block (1) at the top of the Figure 1. For the AES Shift Rows operation, a byte addressable 32-bit wide 32-bit deep shift register is considered block (2), given the resulting compactness of this solution on Xilinx FPGAs when using the SRL32 LUT mode, as demonstrated in the state of the art (previously known as SRL16). This Shift Register is used to temporarily store and address the State. To allow for a more efficient datapath, a forwarding multiplexer is used to select the data fed to the TBoxes. This block is also used to add Round Keys, as in the case of the CLEFIA computation.

The main computation is performed in the TBox (4), used to store the lookup tables that process the byte substitution and the coefficient multiplications. Both AES and CLEFIA require SBox operations followed by a matrix multiplication, which can be partially compressed into a TBox. Depending on the addressed memory region different SBox permutations are performed, thus implementing the AES or CLEFIA operations. The remaining part of the matrices multiplication, the necessary GF(28) additions, are performed by a XOR tree stage (block (5)).

V. Results & analysis

In order to properly compare with the existing state of the art structures, experimental results for Xilinx VIRTEX 5 technology are herein considered, as depicted in Table 1 and Table 2, for AES and CLEFIA algorithms respectively.

The presented values depict the throughput when processing one input block at a time. In unrolled or highly pipelined structures higher throughputs can be achieved, but only if multiple blocks are processed simultaneously. When considering a single data stream in a feedback mode, such as CBC, these structures cannot be efficiently used due to the data dependency between blocks. This implies that the computation of input block *'i'* can only be started after the computation of the input data block *'i-1'* is concluded. As an efficiency metric, we consider the use of the Throughput per Slice ratio [14].

Regarding the key expansion, two main approaches are used, either computing them locally with dedicated logic, or computing them off chip and then storing them in local memory. Although being possible to share some resources with the datapath to perform the key expansion computation, dedicated logic is still required. The off chip computation of the key expansion and loading to an auxiliary BRAM typically yields in more compact and efficient designs, [18].

Since the proposed structure [13] can compute both AES and CLEFIA algorithms, it can be found in Tables 1 and 2. This structure allows for a ciphering throughput of 1 Gbps for the CLEFIA algorithm and near 850Mbps for the AES algorithm. These results are achieved at a cost of 123 Slices and 3 BRAMs on a Virtex 5 device, including the control unit and an extra BRAM for the round keys storage.

Considering the related AES state of the art, the unrolled architecture proposed in [22] allows for a throughput above 60 Gbps in ECB mode. However, when considering feedback modes, considered more secure, the maximum throughput is of 3.2 Gbps with an area cost of 3121 Slices resulting in a Throughput per Slice efficiency of 1.03. The 128-bit folded datapath structure, single cycle per round, presented in [14] achieves a throughput of 2.4 Gbps with an efficiency of 5.96 at a cost of a higher BRAM usage. A more compact structure is proposed in [15] allowing for a throughput up to 1.76 Gbps requiring 107 Slices. However, if feedback modes are used, the maximum throughput lowers to 880 Mbps. An important characteristic of this proposal is the use of 4 DSP blocks to implement the XOR operations, instead of regular Slices. With this option an efficiency of 8.22 Throughput per Slice is achieved. Note that DSPs are Xilinx FPGA dedicated components, and are also not considered in the efficiency metric herein applied. Without the use of DSPs, 212 Slices are needed instead, resulting in an efficiency of 4.15 when considering feedback modes.

When considering the CLEFIA state of the art, the dedicated unrolled structure of [19] allows for a throughput of 21 Gbps in non-feedback modes.

TABLE I. AES IMPLELEMATION SYNTHESIS RESULTS

Results	**Round Structure**	**Device**	**Resources**		**Throughput** [Gbps]	**Efficiency** [Mbps/S]
Designs			*Slices*	*BRAMs*		
Resende et al. [2015]	Rolled (32b)	V5	123	2+1	0.850	6.91
Chaves et al. [2006]	Rolled (32b)	V5	407	8+2	2.427	5.96
Drimer et al. [2009]	Rolled (32b)	V5	107	2+1	0.88	8.22
			212	2+1	0.88	4.15
Liu et al. [2013]	Unrolled	V5	3579	0	2.305	0.64
Bulens et al. [2008]	Rolled (128b)	V5	400	0	1.07	2.67

TABLE II. CLEFIA IMPLELEMATION SYNTHESIS RESULTS

Results	**Round Structure**	**Device**	**Resources**		**Throughput** [Gbps]	**Efficiency** [Mbps/S]
Designs			*Slices*	*BRAMs*		
Kryjak et al. [2009]	Unrolled	V5	2479	0	1.188	0.48
Proença et al. [2011]	Rolled (128b)	V5	170	4+1	1.707	10.04
	Rolled (32b)		86	2+1	1.301	15.13
Resende et al. [2015]	Rolled (32b)	V5	123	2+1	1.073	8.72
		V6	115		1.012	8.80

In feedback modes the maximum throughput is reduces to 1.2 Gbps. With an area cost of 2479 Slices, an efficiency of 0.48 is achieved for feedback modes. The structure proposed in [20] allows for a throughput of 1.7 Gbps at a cost of 170 Slices. The same authors also proposed a more compact structure, allowing for a throughput of 1.3 Gbps at a cost of 86 Slices, resulting in efficiency of 15.13. With a maximum CLEFIA throughput of 1 Gbps.

Overall, the proposed structure allows for a throughput between 1 Gbps and 850 Mbps in feedback modes for CLEFIA and AES algorithms, respectively, at a cost of 123 Slices and 3 BRAMs. The resulting efficiency metric is better than most of the state of the art, supporting only the CLEFIA or the AES ciphers.

VI. FUTURE PERSPETIVES: SECURITY ORIENTED CODES

Cryptographic components as well as on-chip memories are threatened by Differential Fault Analysis (DFA) attacks. DFA attacks use information obtained by examining the difference between the correct operation of a device and its operation in the presence of a fault, in order to retrieve secret or personal information stored in the device. To manipulate the device, an attacker can inject faults and errors of almost any multiplicity and type. Consequently, to protect the device from malicious attacks, all injected errors must be detected with high probability regardless of their multiplicity.

Fault injection attacks can be detected with relatively high probability by error detecting codes. Traditional error detection methods are based on linear codes. Linear codes can detect any random errors of small multiplicity and thus they increase the reliability of hardware systems. However, linear codes cannot increase the immunity of a system against fault injection attacks, in which an attacker can flip any number of bits he wishes [23-24]. Codes that can detect any attack are called security oriented codes.

In general, there are two types of security oriented codes: codes that can detect weak attacks in which the attacker cannot control the codeword to be used, and codes designed to detect strong attacks in which the attacker chooses the information word to be transmitted. Robust codes, such as the Quadratic-Sum and the Punctured-Cubic codes, with or without pre-mapping [25-27] are considered as a countermeasure against weak attacks, and Algebraic Manipulation Detection (AMD) codes [28] are considered a countermeasure against strong attacks.

We distinguish between three types of injected errors: errors that are always detected, errors that are never detected, and errors that are detected with some probability. Fig. 2)a illustrates how an error distorts a weak attack detecting code C. Errors that are always detected map all the codewords to non codewords (see error e_1 in Fig. 2)a).

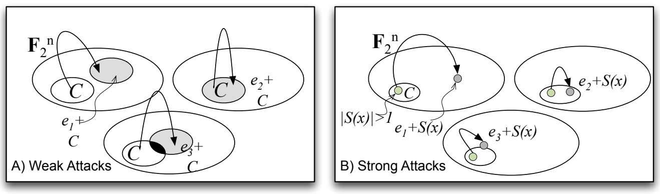

Fig. 2. Error Types for A) Weak and B) Strong Attacks, Detecting Codes

Errors that are never detected map all the codewords onto the code (see error e_2, Fig. 2)a), and errors that are detected with probability map at least one codeword to a non codeword and at least one codeword into the code (error e_3, Fig. 2)a). In a good robust code, such as the Quadratic-Sum and the Punctured-Cubic codes, all the possible errors are detected and the size of the intersection between the code C, and the distorted code (e_3+C) is minimal. In contrast to weak attack detecting code, a code designed against strong attacks must incorporate random bits. That is, an information word, say "x", is randomly encoded into a codeword form a (predefined) set S(x). Fig. 2)b illustrates how an error distorts a strong attack detecting code C. In a good code, there are no undetected errors (e.g. errors like e_2 in Fig. 2)b), and the size of the intersection between the code and the shifted set (e_3+C) is minimal. Strong attack detecting codes, like the AMD codes, are believed to be stronger than robust codes since unlike the last their error masking probability does not depend on the probability-mass-distribution of the codewords. However, when the information is not *uniformly distributed*, i.e. when entropy of the code is small, strong attack detecting codes are not always stronger than the simple low-cost robust codes [29]. A simple criterion, which directs how to choose a security-oriented code with respect to the entropy of the data to be protected, is presented in [29], which can also be applied in modern security schemes integration [30], as well.

ACKNOWLEDGMENT

This work is supported under the framework of EU COST IC 1204: TRUDEVICE (Trustworthy Manufacturing and Utilization of Secure Devices) Project and by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013.

REFERENCES

[1] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "Present: An ultra-lightweight block cipher," in CHES, ser. Lecture Notes in Computer Science, P. Paillier and I. Verbauwhede, Eds., vol. 4727. Springer, 2007.

[2] C. De Canniere, O. Dunkelman, and M. Kneˇ zeviˊ c, "Katan and ktantana family of small and efficient hardware-oriented block ciphers," in Cryptographic Hardware and Embedded Systems-CHES 2009. Springer, 2009, pp. 272–288.

[3] National Institute of Standards and Technology (NIST), "Announcing the Advanced Encryption Standard (AES)," Federal Information Processing Standards Publication 197, November 2001.

[4] T. El-Ghazawi, E. El-Araby, M. Huang, K. Gaj, V. Kindratenko, and D. Buell, "The promise of high-performance reconfigurable computing," IEEE Computer, vol. 41, pp. 69–76, February 2008.

[5] N. Sklavos, "On the Hardware Implementation Cost of Crypto-Processors Architectures", *Information Systems Security, The official journal of (ISC)2, A Taylor & Francis Group Publication*, Vol. 19, Issue: 2, pp. 53-60, 2010.

[6] B. Badrignans, R. Elbaz, and L. Torres, "Secure FPGA configuration architecture preventing system downgrade," in Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on, pp. 317–322, IEEE, September 2008.

[7] H. Kashyap and R. Chaves, "Secure partial dynamic reconfiguration with unsecured external memory," in Field Programmable Logic and Applications (FPL), 2014 24th International Conference on, pp. 1–7, IEEE, 2014.

[8] J. Vliegen, N. Mentens, and I. Verbauwhede, "Secure, remote, dynamic reconfiguration of FPGAs," ACM Trans. Reconfigurable Technol. Syst., vol. 7, pp. 35:1–35:19, Dec. 2014.

[9] F. Devic, L. Torres, J. Crenne, B. Badrignans, and P. Benoit, "SecURe DPR: Secure update preventing replay attacks for dynamic partial reconfiguration," in Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on, pp. 57–62, IEEE, August 2012.

[10] A. J. Elbirt and C. Paar, "An instruction-level distributed processor for symmetric-key cryptography," Parallel and Distributed Systems, IEEE Transactions on, vol. 16, no. 5, pp. 468–480, 2005.

[11] H. Singh, M.-H. Lee, G. Lu, F. J. Kurdahi, N. Bagherzadeh, and E. M. Chaves Filho, "MorphoSys: an integrated reconfigurable system for data-parallel and computation-intensive applications," Computers, IEEE Transactions on, vol. 49, no. 5, pp. 465–481, 2000.

[12] N. Sklavos, P. Kitsos, E. Alexopoulos, O. Koufopavlou, "Open Mobile Alliance (OMA) Security Layer: Architecture Implementation and Performance Evaluation of the Integrity Unit", *New Generation Computing: Computing Paradigms and Computational Intelligence, Springer-Verlag*, Vol. 23, No 1, pp. 77-100, 2005.

[13] J. C. Resende and R. Chaves, "Dual CLEFIA/AES cipher core on FPGA," in Applied Reconfigurable Computing, pp. 229–240, Springer, 2015.

[14] 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, IEEE, 2006.

[15] P. Chodowiec and K. Gaj, "Very compact FPGA implementation of the AES algorithm," in Cryptographic Hardware and Embedded Systems-CHES 2003, pp. 319–333, Springer, 2003.

[16] M. El Maraghy, S. Hesham, and M. A. Abd El Ghany, "Real-time efficient FPGA implementation of AES algorithm," in SOC Conference (SOCC), 2013 IEEE 26th International, pp. 203–208, IEEE, 2013.

[17] G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, and J. Legat, "Compact and efficient encryption/decryption module for FPGA implementation of the AES Rijndael very well suited for small embedded applications," in Information Technology: Coding and Computing, 2004. Proc. ITCC 2004. International Conference on, vol. 2, pp. 583–587, IEEE, 2004.

[18] N. Sklavos and O. Koufopavlou, "Architectures and VLSI implementations of the AES-proposal Rijndael," Computers, IEEE Transactions on, vol. 51, no. 12, pp. 1454–1459, 2002.

[19] T. Kryjak and M. Gorgon, "Pipeline implementation of the 128-bit block cipher CLEFIA in FPGA," in Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on, IEEE, 2009.

[20] R. Chaves, "Compact CLEFIA implementation on FPGAs," in Embedded Systems Design with FPGAs, pp. 225–243, Springer, 2013.

[21] P. Proença and R. Chaves, "Compact CLEFIA Implementation on FPGAs," in Field Programmable Logic and Applications (FPL), 2011 International Conference on, pp. 512–517, IEEE, 2011.

[22] Q. Liu, Z. Xu, and Y. Yuan, "A 66.1 Gbps single-pipeline AES on FPGA," in Field-Programmable Technology (FPT), 2013 International Conference on, pp. 378–381, IEEE, 2013.

[23] V. Tomashevich, S. Srinivasan, F. Foerg, and I. Polian, "Cross-level protection of circuits against faults and malicious attacks," in Proc. IEEE 18th IOLTS, Sitges, Spain, 2012, pp. 150–155.

[24] Victor Tomashevich, Yaara Neumeier, Raghavan Kumar, Osnat Keren and Ilia Polian, "Protecting Cryptographic Hardware against Malicious Attacks by Nonlinear Robust Codes", The IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'14), Amsterdam, 1-3 October 2014.

[25] K. J. Kulikowski, M. G. Karpovsky, and A. Taubin, "Robust codes and robust, fault tolerant architectures of the advanced encryption standard," J. Syst. Archit., vol. 53, no. 2/3, pp. 138–149, Feb. 2007.

[26] Y. Neumeier and O. Keren, "Robust generalized punctured cubic codes," IEEE Trans. Inf. Theory, vol. 60, no. 5, pp. 1–10, May 2014.

[27] Shumsky, O. Keren, and M. Karpovsky, "Robustness of security oriented codes under non-uniform distribution of codewords," in Proc. DSN-DCCS, 2013, pp. 25–30.

[28] M. G. Karpovsky and Z. Wang, "Design of strongly secure communication and computation channels by nonlinear error detecting codes," IEEE Trans. Comput., vol. 63, no. 11, pp. 2716–2729, 2014.

[29] O. Keren and M. Karpovsky, "Relations between the Entropy of a Source and the Error Masking Probability for Security Oriented Codes", IEEE trans. On Communications, Vol. 63, No. 1, pp. 206-214, 2015.

[30] N. Sklavos, "Securing Communication Devices via Physical Unclonable Functions (PUFs)", Information Security Solutions Europe (isse'13), Belgium, 22-23 October, pp. 253-261, Springer, ISBN: 973-3-658-03370-5, 2013.