International Journal of Future Generation Communication and Networking
Vol. 10, No. 12 (2017), pp.71-78
http://dx.doi.org/10.14257/ijfgcn.2017.10.12.07

FPGA Implementation of DLP-PHOTON Hash Function

Baraa Tareq Hammad[1], Yasir Amer Abbas[2], Norziana Jamil[1], Mohd Ezanee Rusli[1] and Muhammad Reza Z`aba[3]

[1]Institute of Informatics and Computing in Energy, Universiti Tenaga Nasional, Malaysia, omrami82@yahoo.com, {Norziana, ezanee}@uniten.edu.my
[2]College of Engineering, University of Diyala, Baquba, Diyala, Iraq yasiramerabbas@gmail.com
[3]MIMOS Berhad, Malaysia, reza.zaba@mimos.my

Abstract

Ensuring the security of resource constrained devices is currently one of the major challenges in cryptography which is commonly addressed by employing lightweight cryptography. In this article, we present the DLP-PHOTON lightweight hash-function suitable for constrained devices such as passive RFID tags. We used a double length sponge construction as domain extension algorithm instead of employing a sponge construction in order to obtain a higher level of security. Double length sponge offers interesting trade-offs between speed, security and hardware. We implement DLP-PHOTON with Spartan 3, Virtex 403 and Artix-7 FPGA boards in an optimized hardware architecture. Our finding shows that the execution speed of DLP-PHOTON light weight hash function is high on a Field Programmable Gate Arrays (FPGA).

Keywords: *Lightweight cryptographic, Hash Function, PHOTON, Sponge construction, DLP-Sponge, FPGA*

1. Introduction

Nowadays, most of the technologies appear in small and compact forms. Some examples are wireless sensors and RFID devices which are used to perform important tasks such as monitoring and gathering data, and sensing environmental parameters in which all the data are sent over wireless network. These devices, which are commonly operated by batteries, are limited in terms of memory space and processing capability. These devices are known as resource constrained devices [1]. To achieve data security/protection, three main security objectives are identified, *i.e.* confidentiality (encryption), integrity (cryptographic hash function) and availability (firewall and network security).

Traditional cryptography is not suitable to be used in such devices because it involves heavy/complex mathematics and long keys. Therefore, lightweight cryptographic algorithms are more preferred. Following this, many primitives have been proposed in block cipher [2, 3, 4, 5, 6, 7, 8] and hash function [9, 10, 11, 12, 13, 14].

When designing new primitives, the compromise between security, performance and hardware must be balanced. Up to date, many lightweight cryptographic algorithms have been proposed. For cryptographic hash function, most primitives use sponge construction [15] such as PHOTON [9] Spongent [10] and Quark [11]. Sponge construction is able to produce random output, but it is not resistant against multicollision attacks which collision can be found with a complexity $2^{(c+3)/2}$. Therefore, it is interesting to have a new construction for lightweight hash function, as well as developing a new design of hash function that will be more resistant against the generic attacks [16, 17, 18].

Received (June 15, 2017), Review Result (September 23, 2017), Accepted (November 24, 2017)

DLP-Sponge [19] was introduced to solve this issue by doubling the length of sponge construction. In this paper, we propose the instantiation of DLP-Sponge, DLP-PHOTON where PHOTON hash function was employed... The new hardware architecture for DLP-PHOTON has been implemented on different FPGA boards such as Spartan 3, Virtex-403, and Artix 7. The finding from our FPGA implementation shows that almost all variants of DLP-PHOTON give higher throughput compared to their PHOTON counterparts.

In this paper, Section 2 and Section 3 will continuous the descriptions of DLP-Sponge and PHOTON. In Section 4, the details of FPGA implementation of DLP-PHOTON will be given. The results will be discussed.

2. DLP- Sponge Construction

In this Section, the DLP-Sponge reported by [19] is briefed. The DLP-Sponge construction uses two *b*-bit compression functions *f*. Two strings are kept similar during the absorbing process. Thereafter, two inputs are taken from the two strings to the compression function *f* as *r* and *c* for the issuance of the output as shown in Figure 1. The procedures of implementing the DLP-Sponge construction are described below.

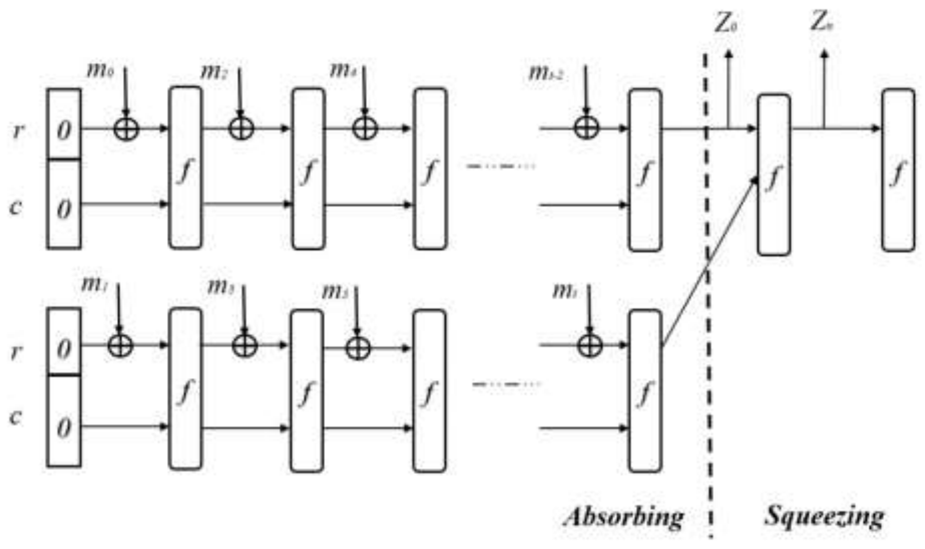

Figure 1. DLP-Sponge Construction

1- The message *M* is padded by *1* bit and *0* bit whereby the length of padded *M* must be a multiple of *r*-bit.

2- Initialize the capacity *c* and the bitrate *r* with zeros.

3- Divide *M* into *r*-bit blocks, $M_0 \ldots M_n$, and process every block of messages sequentially.

4- The absorbing phase: Each input block m_i is XORed with the *r* part of internal state *S*. The DLP- sponge function has an internal state $S = (S_r, S_c) \in 2^r \times 2^c$, where the initial value is $(0, 0)$. Here, $0 \in 2^r$ is the neutral element of *r* and $0 \in 2^c$. S_0 *and* S_1 are iterated until all blocks are exhausted, where S_0 is the upper pipeline state *and* S_1 is the bottom pipeline state.

5- The squeezing phase: The state progresses based on S_f; however, the output block consists of results of the *r* parts of the states obtained after each iteration. The squeezing phase is denoted by: $Z= S_r$, and $S = S_f(S)$.

2. PHOTON

The PHOTON lightweight hash function was designed by Guo *et al.* [9]. It uses a sponge-like construction and an AES-like primitive as internal unkeyed permutation. The output size is $64 \leq n \leq 256$, and the input and output bit rates are r and $r`$. Thus, a PHOTON hash function can be characterized as PHOTON-$n/r/r`$. Internal permutation t with size ($t=c+r$), where r and c are bitrate and capacity, respectively. There are five different versions with unique digest sizes: 80, 128, 160, 224 and 256 bits. The process involves two phases, *i.e.* absorbing and squeezing. First, the input message is padded and divided into m blocks of size r-bit. Then, the absorbing phase is started by XORing the incoming m input into r-bit. After performing absorption on the message block, one applies a t-bit permutation P on the internal state, all the parameters of PHOTON has been shown in Table 1. Once all message blocks have been processed, the squeezing phase starts. During this phase, for each iteration, $r`$ bits are generated from the internal state and the permutation P is applied. Squeezing continues until the proper digest size n is reached.

The PHOTON internal permutation P consists of 12 rounds. The internal state is represented by ($d \times d$) matrix of s-bit cells and each round involves four operations: AddConstants, SubCells, ShiftRows, and MixColumnsSerial as shown in Figure 2.

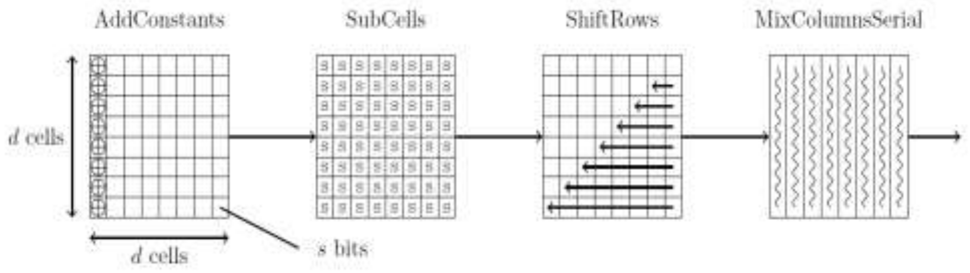

Figure 2. One Round of a PHOTON Permutation

Table 1. The parameters of the internal permutations Pt, together with the internal constants ICd, the irreducible polynomials and the Zi coefficients for the MixColumnsSerial computation.

	t	*d*	*s*	*Nr*	***IC**$_d$* **(.)**	***irr. polynomial***	***Z**$_i$* ***coefficients***
***P**$_{100}$*	100	5	4	12	[0, 1, 3, 6, 4]	$x^4 + x + 1$	(1, 2, 9, 9, 2)
***P**$_{144}$*	144	6	4	12	[0, 1, 3, 7, 6, 4]	$x^4 + x + 1$	(1, 2, 8, 5, 8, 2)
***P**$_{196}$*	196	7	4	12	[0, 1, 2, 5, 3, 6, 4]	$x^4 + x + 1$	(1, 4, 6, 1, 1,6, 4)
***P**$_{256}$*	256	8	4	12	[0,1,3,7,15,14,12,8]	$x^4 + x + 1$	(2, 4, 2, 11,2,8,5,6)
***P**$_{288}$*	288	6	8	12	[0, 1, 3, 7, 6, 4]	$x^8 + x^4 + x^3 + x + 1$	(2, 3, 1, 2, 1, 4)

3. FPGA Implementation

The new DLP-PHOTON was designed based on DLP-Sponge and PHOTON-80/20/16. The implementation of new FPGA involves the following steps:

i. Designing the hardware architecture and the data flow of DLP-PHOTON as shown in Figure 3.

ii. Writing the VHDL codes for DLP-PHOTON.

iii. The VHDL code is synthesized, translated, mapped, placed, and simulated using Xilinx ISE design suite 14.5. The new hardware design involves controlling the data flow through two multiplexers and state machines.

The main VHDL components used in the implementation of DLP-Sponge construction are shown in Figure 4. The input data must be padded. Then, the absorbing phase will work in two parallel lines. For each line, a 2×1 multiplexer drives r bits of the data input from message registers and applies the XOR operation with IVr. After the padding procedure, this multiplexer operates as a feedback multiplexer in order to apply the 12 rounds of internal permutation of PHOTON. The data register is updated every round after processing AddConstants, SubCells, ShiftRows, and MixColumns. Another 2×1 multiplexer is used to drive either the IV value or the internal state. Finally, during the squeezing phase, r bits are generated from the internal state after each permutation P until the hash digest of length n is obtained.

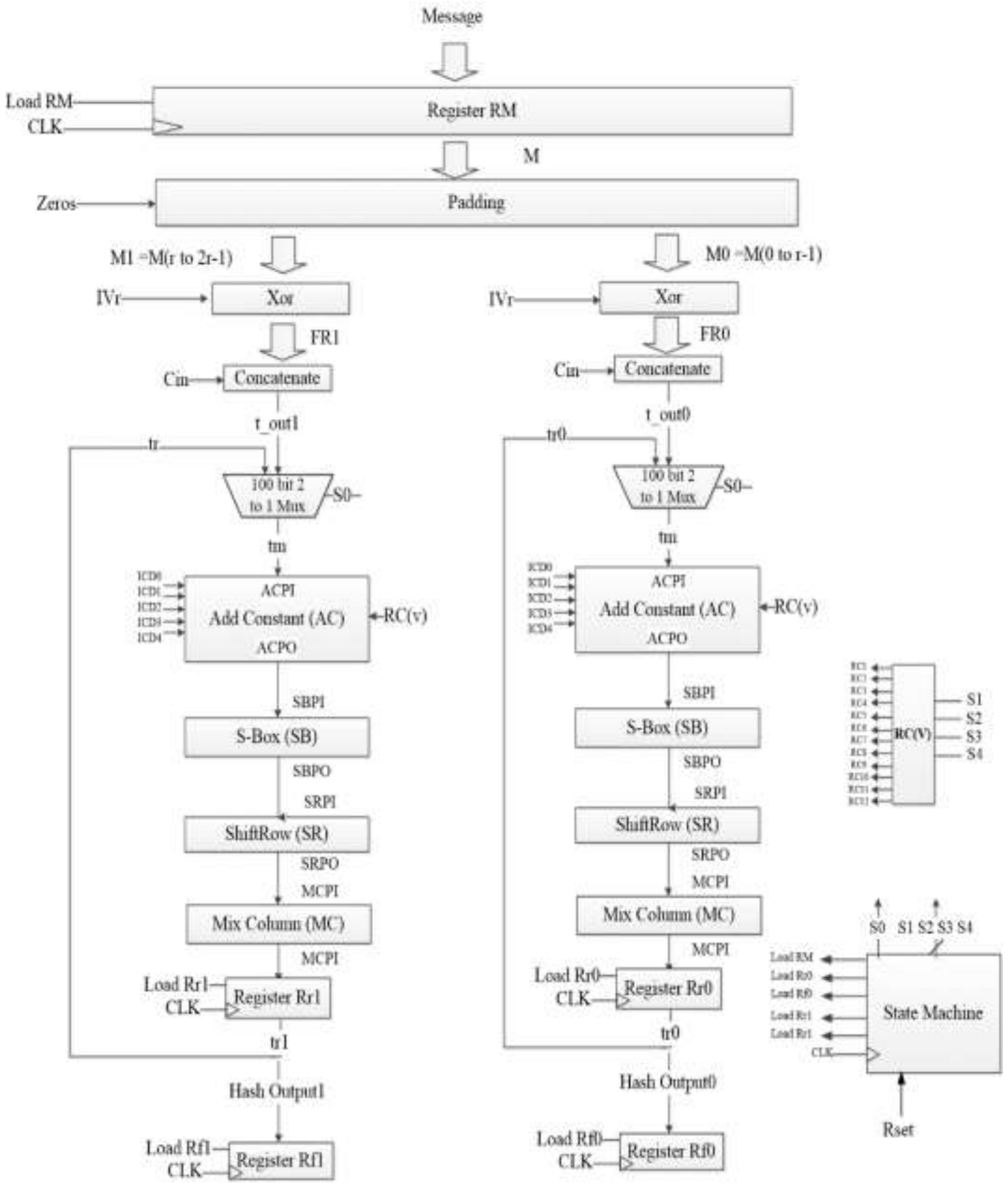

Figure 3. The Data Flow of the DLP-PHOTON

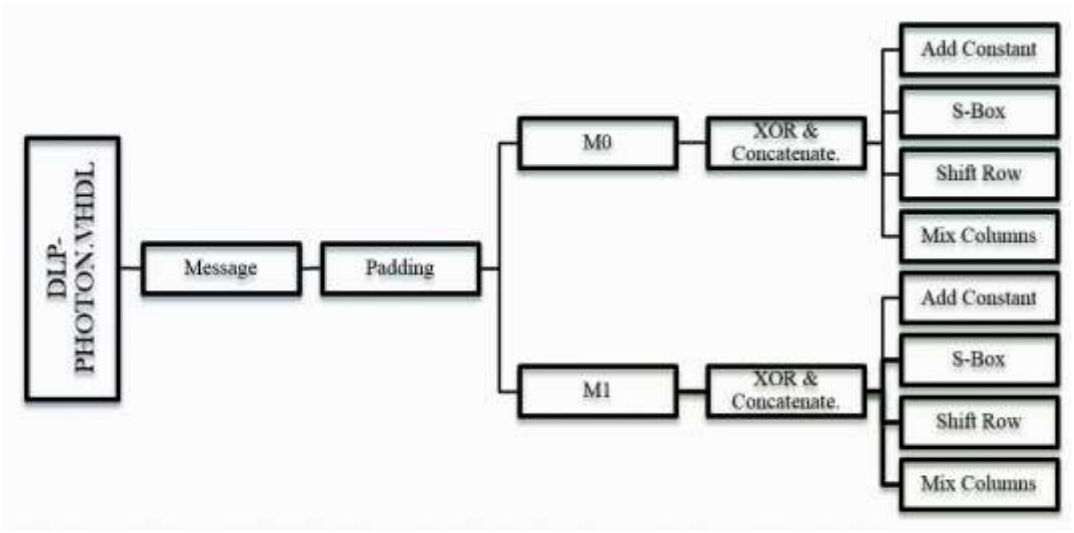

Figure 4. The VHDL Implementation of the DLP-PHOTON

4. Result and Analysis

The main target of our design is to improve the efficiency of PHOTON via implementing the new DLP-PHOTON. It is challenging to optimize all parameters such as security, hardware and efficiency [20] at the same time. For example, by using the pipelined, side-channel-resistant architecture in our design, the performance is undoubtedly high. However it incurs a relatively high cost as well.

Figure 5, shows the test vector for input and output in simulation graph. The state machine will be controlled by the hardware design for DLP-PHOTON hash function. The figure shows the simulation graph for the message input to DLP-PHOTON hardware designed then after 12 clock cycles the two output results will be executed in parallel.

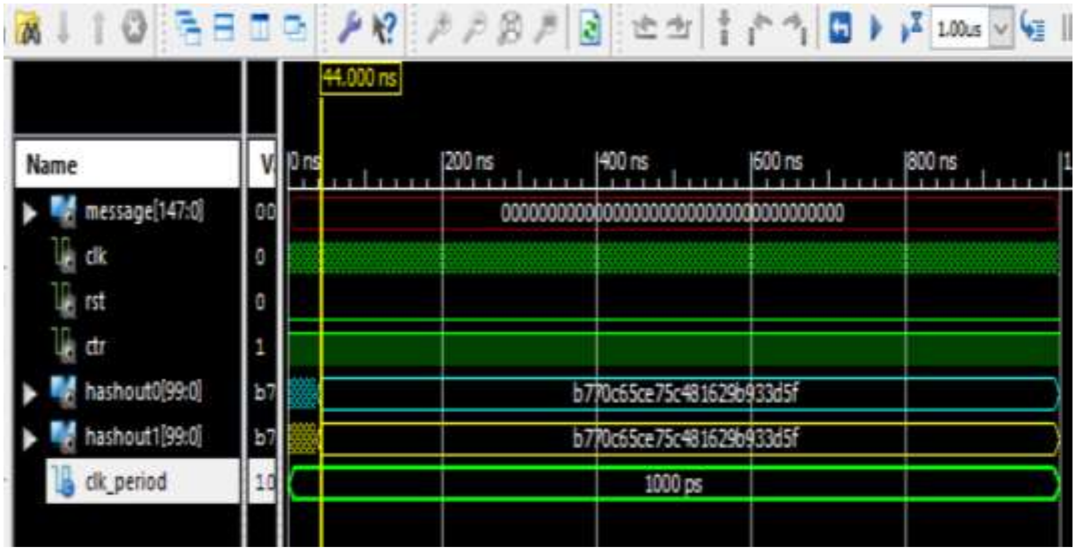

Figure 5. The Simulation of DLP-PHOTON

The proposed DLP-PHOTON design was successfully synthesized using three Xilinx FPGA boards. Table 2 shows the results of DLP-PHOTON with Spartan 3, Virtex 403 and Artix-7 FPGA boards. Designing DLP-PHOTON using different FPGA boards gives numerous values of slice number, frequency, throughput and efficiency.

Table 2. Proposal Design of DLP-PHOTON with Different FPGA Boards

	FPGA	Data	CLK	Slices	Freq.	Throughput	Efficiency
DLP-PHOTON	Spartan 3	100	12	615	308	1027	1.67
DLP-PHOTON	Virtex-403	100	12	815	594	1980	2.42
DLP-PHOTON	Artix-7	100	12	402	903	3010	7.48

The proposed DLP-PHOTON hardware design was then compared with the existing PHOTON FPGA design in two platforms Spartan-3 and Artix-7. The results presented in Table 3 show that the frequency, speed, throughput and efficiency of DLP-PHOTON are higher than those of other PHOTON counterpart implemented on FPGA.

Table 3. Comparison DLP-PHOTON with existing PHOTON FPGA Implementations

Algorithm	FPGA Board	Data	CLK	Slices	Freq. (MHz)	Throughput (Mbps)	Efficiency (Mbps/Slices)
DLP-PHOTON Proposed	Spartan-3	100	12	615	308	1027	1.67
PHOTON (80/20/16) [21]	Spartan-3	100	12	285	78.53	130.88	0.46
DLP- PHOTON Proposed	Artix-7	100	12	402	903	3010	7.48
PHOTON (80/20/16)[21]	Artix-7	100	12	142	232.65	387.75	2.73

5. Conclusion

In this chapter we propose the instantiation of the DLP-Sponge in a form of a new cryptographic hash function. We employ the existing PHOTON lightweight hash function to construct DLP-PHOTON. The new DLP-PHOTON designed architecture was successfully synthesized, mapped, simulated and tested on an FPGA evaluation board. The results show that the proposed hardware architecture achieves significant improvements throughput to 3.01 Gbps and increasing the efficiency to 7.48 Mbps/slices with Artix 7 FPGA board The use of DLP-Sponge also offers interesting trade-offs between area, speed and security.

Acknowledgement

This research is partly funded by TNB Seed Fund 2016 under a project 'RandD in Micro Cryptographic Engine for Protecting Data in SCADA Link'.

References

[1] A.M. Deshpande, M.S. Deshpande and D.N. Kayatanavar, “FPGA implementation of AES encryption and decryption”, Proceeding of International Conference on Control, Automation, Communication and Energy Conservation (INCACEC, 2009), **(2009)**, pp. 1-6.

[2] A. Bogdanov, L. R.K nudsen, G. Leander, C. Paar, A. Poschmann, M. J.Robshaw and C. Vikkelsoe, “PRESENT: An ultra-lightweight block cipher”, In International Workshop on Cryptographic Hardware and Embedded Systems (pp. 450-466). Springer Berlin Heidelberg, **(2007)**.

[3] W. Wu and L. Zhang, “LBlock: a lightweight block cipher. In International Conference on Applied Cryptography and Network Security”, Springer Berlin Heidelberg, **(2011)**, pp. 327-344.

[4] C. H. Lim and T. Korkishko, “mCrypton–a lightweight block cipher for security of low-cost RFID tags and sensors”, In International Workshop on Information Security Applications, Springer Berlin Heidelberg, **(2005)**, pp. 243-258.

[5] M. Hamann, M. Krause and W. Meier, “LIZARD–A Lightweight Stream Cipher for Power-constrained Devices”, IACR Transactions on Symmetric Cryptology, vol. 2017, no. 1, **(2017)**, pp. 45-79.

[6] A. Ali, “Oppel-1: A new block cipher”, In Applied Sciences and Technology (IBCAST), 2017 14th International Bhurban Conference on, IEEE, **(2017)**, pp. 441-447.

[7] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B. S. Koo and H. Kim, “HIGHT: A new block cipher suitable for low-resource device”, In International Workshop on Cryptographic Hardware and Embedded Systems, Springer Berlin Heidelberg, **(2006)**, pp. 46-59.

[8] M. R. Z’aba, N. Jamil, M. E. Rusli, M. Z. Jamaludin and A. A. M. Yasir, “I-PRESENT TM: An Involutive Lightweight Block Cipher”, Journal of Information Security, vol. 2014, **(2014)**.

[9] J. Guo, T. Peyrin and A. Poschmann, “The PHOTON family of lightweight hash functions”, In Annual Cryptology Conference, Springer Berlin Heidelberg, **(2011)**, pp. 222-239.

[10] A. Bogdanov, M. Knežević, G. Leander, D. Toz, K. Varıcı and I. Verbauwhede, “SPONGENT: A lightweight hash function”, In International Workshop on Cryptographic Hardware and Embedded Systems, Springer Berlin Heidelberg, **(2011)**, pp. 312-325.

[11] J. P. Aumasson, L. Henzen, W. Meier and M. Naya-Plasencia, “Quark: A lightweight hash”, In International Workshop on Cryptographic Hardware and Embedded Systems, Springer Berlin Heidelberg, **(2010)**, pp. 1-15.

[12] T. Berger, J. D’Hayer, K. Marquet, M. Minier and G. Thomas, “The GLUON family: a lightweight hash function family based on FCSRs”, Progress in Cryptology-AFRICACRYPT 2012, **(2012)**, p. 306-323.

[13] E. B. Kavun and T. Yalcin, “A lightweight implementation of Keccak hash function for radio-frequency identification applications”, In International Workshop on Radio Frequency Identification: Security and Privacy Issues, Springer Berlin Heidelberg, **(2010)**, pp. 258-269.

[14] P. M. Mukundan, S. Manayankath, C. Srinivasan and M. Sethumadhavan, “Hash-One: a lightweight cryptographic hash function”, IET Information Security, vol. 10, no. 5, **(2016)**, pp. 225-231.

[15] G. Bertoni, J. Daemen, M. Peeters and G. Van Assche, Cryptographic sponges. online] http://sponge. noekeon. org, **(2011)**.

[16] G. Bertoni, J. Daemen, M. Peeters and G. Van Assche, “Sponge functions”, In ECRYPT hash workshop, vol. 2007, **(2007)**.

[17] M.A. Alahmad, I.F. Alshaikhli and M. Nandi, “Multicollisions in Sponge construction”, In Informatics and Creative Multimedia (ICICM), 2013 International Conference on, IEEE, **(2013)**, pp. 215-219.

[18] B. T. Hammad, N. Jamil, M.E. Rusli and M. Reza Zaba, “Faster multicollision attack on Sponge Construction”, advanced science letters, in press, **(2017)**.

[19] B. T. Hammad, N. Jamil, M.E. Rusli and M. Reza Zaba, “DLP sponge construction for authenticated encryption in Zulikha, J. and N. H. Zakaria (Eds.)”, Proceedings of the 6th International Conference on Computing and Informatics Sintok: School of Computing, **(2017)**, pp. 714-721.

[20] H. Mobahat, “Authentication and lightweight cryptography in low cost RFID”, In Software Technology and Engineering (ICSTE), 2010 2nd International Conference on, IEEE, vol. 2, **(2010)**, pp. V2-123.

[21] N. N. Anandakumar, T. Peyrin and A. Poschmann, “A Very Compact FPGA Implementation of LED and PHOTON”, Progress in Cryptology--INDOCRYPT 2014, **(2014)**, pp. 304–321.