Journal of Scientific & Industrial Research
Vol. 82, March 2023, pp. 316-321
DOI: 10.56042/jsir.v82i03.71759

DeeR-Gen: A Pseudo Random Number Generator for Industry 4.0 / IoT

Deena Nath Gupta* & Rajendra Kumar
Jamia Millia Islamia, New Delhi 110 025, India

Received 02 August 2022; revised 18 September 2022; accepted 06 October2022

Random binary bit sequences or random numbers are very useful in cryptographic applications. These sequences are used as a key in different encryption algorithms. Also, they can be used as random nonce in many mutual authentication protocols. Because these sequences are used at very basic level in cryptographic applications there generation should be fast, secure, and energy-efficient. Particularly in the case of Industry 4.0/IoT, a lightweight implementation is much needed along with high security and rapid production. The earlier generators of random numbers used the true source of randomness but the same is not feasible in current scalable Industry 4.0/IoT scenario. Many works have already been done to generate random numbers through PRNGs. Some examples are J3Gen, Warbler, LAMED, and ARROW. However, it is essential to bring a completely programmed, highly secured, energy efficient and a fast paced algorithm for random number generation. In this paper, a novel algorithm, named DeeR-Gen, which works with one multiplexer and two NLFSRs is presented. It requires only 245 GE on ASIC, lowest hardware requirement till date. Proposed methodology has also been tested for EPC test of randomness. The authors found the proposed algorithm secure and energy-efficient to be used in any lightweight cryptographic algorithm.

Keywords: IoT, Industry 4.0, Lightweight cryptography, Random nonce, Secret key

Introduction

The communications between constrained devices needs to be secure so that more applications can adopt Industry 4.0/IoT environment. One of the basic requirements of security mechanisms is random number. The security mechanisms use the random numbers in many ways. Random numbers can be used as a secret code as well as random bit sequences can also be used as a key for encryption. These random numbers can be generated from True Random Number Generators (TRNGs) or Pseudo Random Number Generators (PRNGs). Both of them are having their strengths. TRNGs are true source of randomness that produces highly random outputs while PRNGs are independent of any outside source and hence can produces random numbers in a fast pace.[1–3]

Many researchers were replicating the output of TRNGs by using different software methodologies. The first such method was presented by Blum BlumShub in 1986 into their article named "A Simple Unpredictable Pseudo Random Number Generator." Other proposals include the generation of random numbers by using the concept of genetic programming, primitive polynomials, circuit changing their state from meta-stable to bi-stable and Mersenne twister. All the above concepts were successful to generate random numbers as per the requirement of cryptographic application. However, the current cryptographic requirements are somewhat different from the previous one seeing the lightweight implementation scenario.[4–7]

In Industry 4.0/IoT, almost 80% of the total communications are happening between constrained devices only. Hence, the needs and requirements of constrained devices must be taken care of. Due to their low energy design, constrained devices are not able to run highly complex programs. This gives birth to the requirement of lightweight cryptography.[8] To generate random binary sequences that contain low power to run is challenging. To cop up with this challenge the mathematicians suggested using the shift operations. The shift operations take the lowest CPU power among all other logical operations. Hence, many research starts including shift operations into their design. Some of the examples are, J3Gen, Melia Segui, Warber, LAMED, and ARROW.[9–12]

Shift registers can be implemented by using FPGA (Field Programmable Gate Arrays). It can be seen that some of the designs used the concept of LFERs (Linear Feedback Shift Registers) while others used the concept of NLFSRs (Non-Linear Feedback Shift Registers).[13] To found a perfect tradeoff between power usage and security, many of the researchers

*Author for Correspondence
E-mail: prof.dev.cse@gmail.com

started using a combination of LFSRs and NLFSRs in their design. These LFSRs/NLFSRs were then seeded to get the initial values for computation. Many authors use the primitive polynomials of required degree, i.e., polynomial of degree 16 or polynomial of degree 32, or any other required degree of polynomial to seed their shift registers. An update function is then required to get the input value based on the FSR values for the next iteration.

Many researchers used the polynomials directly while some others used the concept of selection criteria for selecting particular polynomial for seeding the FSRs. The direct use of polynomial is not secured because any attacker can have the values from the applied polynomial and can guess the outputs easily.[14] On contrary, polynomial selection from pool of polynomials based on the values received from selection wheels is hard to guess. For example, the proposed mechanism is using a list of eight polynomials and a selection criterion is applied on them to choose one polynomial for the seeding purpose. In this way, the chance of knowing the values from selected polynomial is reduced to eight folds. Different researchers used different mechanism for the selection operation on polynomials. Some of the examples are roulette wheel, random selection, and circuit switch based selection mechanism.

In the presented article, the authors proposed a pseudo random binary bit sequence generator based on one polynomial selection mechanism (multiplexer) and two NLFSRs. The proposed architecture produces secure random binary bit sequences to be used in cryptographic applications. The generator will be fast enough to produce the required amount of random numbers for a scalable Industry 4.0/IoT application. The proposed algorithm is tested for its power consumption and the authors found that it requires vary low CPU power to execute.

Related Work

LAMED architecture for PRNG is proposed in 2007 which is based on an Initialization Vector (IV) and a key.[9] The update function used in LAMED is different for odd length sequences and even length sequences. LAMED was originally proposed as 32-bits sequence generator. However keeping the specifications of EPC in mind, an 16-bits sequence generator is proposed and referred as LAMED-EPC. Another design is proposed a PRNG in 2008 based on simple LFSR.[14] Although this was the first PRNG of this kind, it was not that much successful because of its inherent linearity.

In 2011, two different architectures for PRNGs were proposed. AKARI was one of them and it is yet another architecture based on a non-linear filter function.[5] Two alternatives were proposed, AKARI-1, and AKARI-2. AKARI-1 iterates its filter function for 64 rounds where AKARI-2 iterates for 24 rounds only. The initialization of x0 and x1 is same for both the versions but there is a slight difference in the calculation of z. Also, the iterative function is different for both the version. In AKARI, the contributors tried to incorporate both the requirements of lightweight cryptography, i.e. low power requirement and low area requirement. However it did not succeed in unified design. AKARI-1 tries to lower the power requirement while AKARI-2 tries to lower the area requirement.

A 16-bit PRNG is proposed with eight different polynomials using three TRN bits, i.e. $2^3 = 8$. The expected Gate Equivalent for the proposed PRNG was 761. It is also tested for suitability of the design with EPC global and found it suitable to be used in lightweight cryptographic algorithms. The randomness test for the generated sequences was also performed and the obtained results were satisfactory.[15–21]

J3Gen is another design for generating pseudo random bit sequences proposed in 2013.[(12)] The design was based on a physical source of true randomness and a deterministic LFSR. A decoding logic is used to select one polynomial out of eight given. This decoding logic works on the input from a TRN bit and a clock input. The lowest GE requirement for J3Gen is 440 for its 16-bits LFSR and 8-bits polynomial version while the highest GE requirement for J3Gen is 3921 for its 64-bits LFSR and 32-bits polynomial version.

Proposed Methodology

The authors proposed a new methodology (Fig. 1) for generating the pseudo random binary bit sequences those will perfectly replicate the true random binary bit sequences. The generated sequences will be suitable to

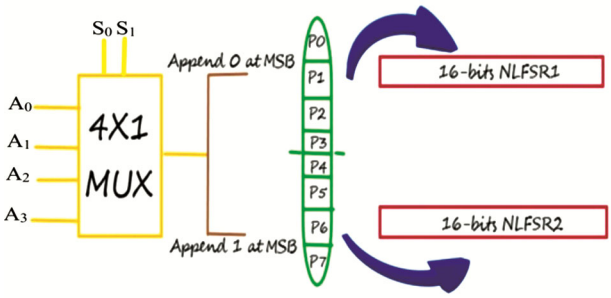

Fig. 1— Selection of polynomial

be used in any cryptographic applications. The authors are using one 4×1 multiplexer along with two 16-bit NLFSRs for their design. At the moment the Tag reaches the periphery of the Reader, the multiplexer inside the Reader will be triggered. It will generate one output from four possibilities. This output (00, 01, 10, or 11) will be appended at MSB with 1 and 0 separately. This will ultimately produce (100, 101, 110, or 111) as input for first NLFSR and (000, 001, 010, or 011) as input for second NLFSR. The respective polynomial from the list of polynomials will be taken as input to both the NLFSRs.

These input numbers will be mapped with a list of polynomials. The respective polynomial will be chosen from the list of polynomial as shown in Table 1. These polynomials are selected from the list of primitive irreducible polynomials of degree 16. Also these NLFSRs will be updated on the basis of feedback functions of 16-bits Fibonacci NLFSRs with the period 2^n-1 respectively as presented in Table 1. The update functions can be chosen from the eight availabilities. The stored update functions are [0, 1, 2, 3, 9, (6, 14)], [0, 1, 5, 13, 14, (14, 15)], [0, 1, 11, 12, 13, (5,15)], [0, 2, 5, 10, 14, (6,14)], [0, 2, 6, 11, 12, (14, 15)], [0, 2, 7, 8, 10, (3, 6)], [0, 2, 7, 8, 13, (3,15)], and [0, 4, 8, 9, 10, (8,12)]. Also, the update function will keep changing on each round. The updated function can be generated as per the given Eq. (1). For example, the update function for {0, 1, 2, 3, 9, (6, 14)} will be:

$$f(X_0,X_1,X_2,X_3,X_6,X_9, X_{14}) = X_o \oplus X_1 \oplus X_2 \oplus X_3 \oplus X_9 \oplus X_6X_{14} \quad \ldots (1)$$

The 16-bits seed in the NLFSR based on the polynomial at input number 000 will be 1000000111101110. Please note that the positions having a power of X are given 1 and the others are given 0 here. The 1 at the last position of each polynomial will be ignored.

Table 1 — List of primitive polynomials with update functions

Input Number	Polynomial
000	x^16 + x^9 + x^8 + x^7 + x^6 + x^4 + x^3 + x^2 + 1
001	x^16 + x^13 + x^12 + x^10 + x^9 + x^7 + x^6 + x^1 + 1
010	x^16 + x^13 + x^12 + x^11 + x^7 + x^6 + x^3 + x^1 + 1
011	x^16 + x^13 + x^12 + x^11 + x^10 + x^6 + x^2 + x^1 + 1
100	x^16 + x^14 + x^13 + x^12 + x^6 + x^5 + x^3 + x^2 + 1
101	x^16 + x^15 + x^10 + x^6 + x^5 + x^3 + x^2 + x^1 + 1
110	x^16 + x^15 + x^11 + x^10 + x^9 + x^6 + x^2 + x^1 + 1
111	x^16 + x^14 + x^13 + x^12 + x^10 + x^7 + 1

After seeding the NLFSRs, the tangled architecture will be followed for generating a random binary digit. The tangled architecture is demonstrated in Fig. 2. The last bit of both the NLFSRs will be XORed and this value will again be XORed with the output of both the update functions separately. The resultant from both the operations will be inserted in both the NLFSRs separately from rear end.

Also, these calculated updates will be XORed again and the resultant will be taken out as output from the PRNG. By repeating the same process 16 times, the methodology will produce a 16-bits long random binary bit sequence. If any application needs longer bit sequences then the polynomials can be chosen accordingly. These sequences can also be used as random numbers/codes by just converting them into their hexadecimal or ACII equivalent.

Results and Discussion

The proposed methodology is programmed in C# language on NET framework. The authors used Intel(R) Core(TM) i3-5005U CPU @ 2 GHz system to analyze their code. The generated random sequences are tested for the suitability with EPC C1 Gen2 randomness test.[17] In the subsequent subsection the results of different tests will be given.

Suitability with EPC Gen2

The requirements of EPC Gen2 for random binary bit sequences are three-fold. The first one is related to the frequency of occurrence of different sequences. Any 16-bits sequences should fall in between the probability as per Eq. (2).

$$\{P_{min} = 0.8/2^{16}\} < Prob(S) < \{P_{max} = 1.25/2^{16}\} \quad \ldots (2)$$

The second requirement is related to duplicate sequences. The same 16-bits sequences should not be repeated for more than 0.1% of times in a generation of 10,000 sequences. The third requirement is related

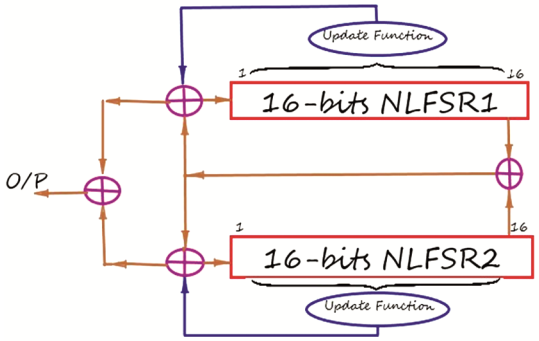

Fig. 2 — Process of generating random bit from tangled NLFSRs

with the guessing of correct sequence. Considering all previous sequences is known to the adversary, the chance of getting the next 16-bits sequence should be less than 0.025%.

The authors tested their code for all three requirements of EPC Gen2. It can be seen that the proposed methodology fulfills every requirement of EPC Gen2 standard. For the first requirement, the authors generated 30 million 16-bits binary sequences. The obtained result shows the fulfillment of Eq. (2). The statistics can be seen from Fig. 3. Many of the researchers stated that the EPC test is not very much suitable for current scenario. Hence, the author takes care of this and tried to produce stronger generator. The results in Fig. (3) show that the proposed generator produces better result than the required one by EPC. It can be seen that as the number of sequences generated grows, the percentage of repetition falls. This result is particularly very beneficial in case of scalable environments, such as IoT. For the second requirement of EPC test, the authors run their code for ten different rounds to generate 10,000 number of 16-bits binary sequences. The obtained results are presented in Table 2. It can be seen in each round that more than 9200 sequences are unique. Only few of the sequences are repeated twice, thrice, or four times. The highest repetition for four times gives a percentage of 0.04% that is too less in comparison to the allowed 0.1% and hence authors claimed that the proposed PRNG is suitable to be used in a cryptographic environment.

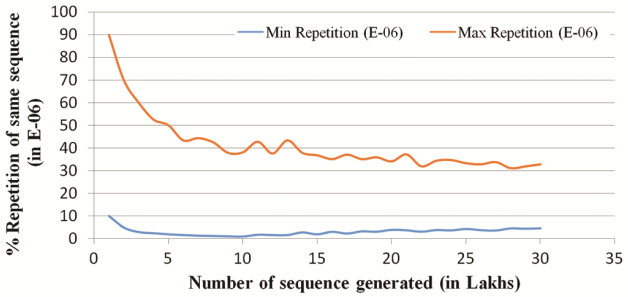

Fig. 3 — Percentage of repetitions of same sequences

For the third requirement, the methodology needs to be revisited. It can be seen that at the very first step it selected one out of four possible inputs leading the probability of correct selection to ¼ and then at the time of update function selection it again choses one out of eight possibilities leading the cumulative probability to 0.0132. This is clearly very less than the allowed 0.025. Hence, is it clear that the proposed methodology pass every test from EPC Gen2 and with this the authors claim that the proposed method is very suitable to be used in a lightweight cryptographic security schemes.

Power Analysis

In cryptographic operations, the energy required depends on the average power (Pavg) and the computation time t. In battery-powered devices, energy consumption is a major parameter which is affected by the time the battery is able to provide electricity. While passively powered devices (such as EPC Gen2 tags) carry only small amounts of power, they are generally capable of connecting to a reader for an extended period of time. In other words, the amount of energy does not matter as long as the calculation can be done within a reasonable amount of time in the EPC Gen2 tags.[18] A large number of digital circuit designs are built using standard CMOS transistors in order to achieve low power consumption and robustness. An implementation using CMOS technology is therefore appropriate for analyzing power consumption.[19]

A key aspect of the design phase of a security implementation is to ensure that the IC's dissipation

Table 2 — Analysis based on 10,000 generated sequences

Round	No. of sequences				Highest repetition (%)	Unique sequences (%)
	Unique	Repeated twice	Repeated thrice	Repeated four times		
1	9267	677	28	0	0.03	92.67
2	9220	718	31	0	0.03	92.20
3	9239	705	25	2	0.04	92.39
4	9229	702	33	1	0.04	92.29
5	9242	667	41	3	0.04	92.42
6	9231	673	45	2	0.04	92.31
7	9238	685	34	3	0.04	92.38
8	9299	644	27	1	0.04	92.99
9	9297	634	33	1	0.04	92.97
10	9244	691	31	1	0.04	92.44

Table 3 — GE comparison of lightweight PRNG proposals

PRNG	Trivium	LAMED	Grain	Melia-Segui	J3Gen	DeeR-Gen
GE Count	1857	1585	1294	453	439	245

does not exceed its power budget for operation. Based on Feldhofer *et al.*'s estimations, cryptographic operations consume approximately 4 μW of power.[18] Dynamic and static power consumptions are added up to calculate the power consumption of CMOS circuits. Due to its small size and low static power consumption, it can be ignored in the circuit design of the proposed implementation. While direct measures of power dissipation may be feasible, a simple method for estimating the dynamic dissipation of power is to calculate the power loss during charge and discharge of capacitances.[19] A system with a small number of logic gates dissipates power as shown in Eq. (3).

$$P = p_{0\to 1} C_L V^2_{DD} f_{clk} \quad \ldots (3)$$

The capacitor CL along the critical path represents the load capacitance and the logic state $p_{0\to 1}$ represents the transition from low to high in one clock cycle. Combined with CL, $p_{0\to 1}$ can also be expressed as the average capacitance switched during each clock cycle. Clock frequency is represented by f_{clk}, and system supply voltage is represented by V_{DD}. The factors in this equation that influence the power consumption are minimized when designing measures to reduce it. For large ICs with many logic states transitions per clock cycle, this formula has difficulty being applied due to difficulties in stating the logic states transitions. Nonetheless, it is an adequate method for estimating the power consumption of small circuits.

It is estimated that the Load Capacitance (CL) for each GE is approximately 3 fF based on measurements presented by Etrog *et al.*[20] Passive low-cost RFID uses a voltage source of 1 V and an operating frequency of 100 kHz, which have been previously stated.[7] For our PRNG proposal that executes in serial, Eq. (1) returns an estimate of 7.35 nW of average power consumption, assuming that all GEs are switched for every clock cycle. Based on existing literature, and given the available budget of 4 μW of power consumption for cryptographic operations for UHF technologies, the proposed estimation is consistent.[21]

Performance Analysis

Implementation of a 4×1 multiplexer can be done with 11 numbers of 2-input NAND Gates and hence required only 11 GE on ASIC. A 16-bits FSR is made up of 16 D flip-flop placed serially. One D flip-flop needs 5 numbers of 2-input NAND Gates and hence required only 5 GE area on ASIC. For a single NLFSR, the design will require 80 GE. Since, this design is having 2 NLFSRs a total of 160 GE will be required for implementing two 16-bits NLFSRs. The update function of a single NLFSR is requiring a total of 18 GE. Hence, two simultaneous update operations on two different LFSRs will require a total of 36 GE. Around 26 GE will be required to implement an 8 × 1 multiplexer for selecting one update function among the eight available. The further design of DeeR-Gen is having four more XOR operations and hence they need 12 more GE. Combining all the values together the total area requirement for the DeeR-Gen design will be (11 + 80 + 80 + 18 + 18 + 26 + 12) = 245 GE. The comparison of different lightweight PRNG proposals is given in Table 3.

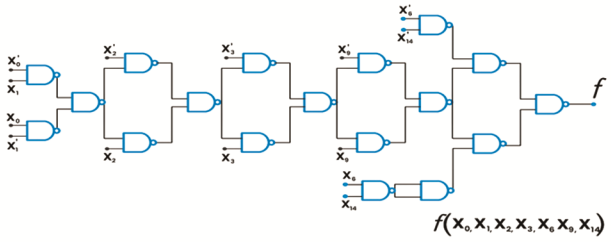

Fig. 4 — Circuit diagram for update function

The number of required gate equivalent for one of the update function in Eq. (1) is illustrated in Fig 4. It is worth noticing that all the update function will take the same number of GE for their execution.

Conclusions

Highly secured yet energy-efficient random number generator is proposed that is efficient both in terms of energy and area requirements. The proposed method, DeeR-Gen, takes only 245 GE on ASIC, lowest area requirement by any of the known PRNG. Also the energy requirement of the digital circuit is 7.35 nW, much lower than the permissible range of 4 μW for constrained environments. In addition to energy and area requirements, the authors also tested the generated sequences for randomness. EPC requirements were executed and it is found that the proposed PRNG passed all the tests. This shows that

the generated sequences are highly suitable to be used in a lightweight cryptographic environment. In future, authors will consider different arrangements of LFSRs and NLFSRs to produce yet another level of secure and lightweight random binary bit sequences. Also, different degree of primitive polynomials will be checked for better seeding of the FSRs.

References

1 Anand R, Sinha A, Bhardwaj A & Sreeraj A, Flawed security of social network of things, in *Handbook of Research on Network Forensics and Analysis Techniques* (IGI Global) 2018, 65–86, https://doi.org/10.4018/978-1-5225-4100-4.ch005.

2 Gupta A, Srivastava A, Anand R & Tomažič T, Business application analytics and the internet of things: The connecting link, In *New Age Analytics* (Apple Academic Press) 2020, 249–273.

3 Gupta R, Shrivastava G, Anand R & Tomažič T, IoT-based privacy control system through android, *In Handbook of E-business Security* (Auerbach Publications) 2018, 341–363

4 Melia-Segui J, Garcia-Alfaro J & Herrera-Joancomarti J, Analysis and Improvement of a Pseudorandom Number Generator for EPC Gen2 Tags, Financial Cryptography and Data Security*: Lect Notes Comput Sci* (**6054)** (2010), https://doi.org/10.1007/978-3-642-14992-4_4.

5 Martin H, San Millan E, Entrena L, Lopez P & Castro J A: A pseudorandom number generator for secure lightweight systems, *IEEE 17th Int on-line Test Symp* (Athens, Greece) 2011), 228–233, https://doi.org10.1109/IOLTS.2011.5994534.

6 Tsoi K H, Leung K H & Leong P H W, Compact FPGA-based true and pseudo random number generators, *11th Proc Annu IEEE Symp Field-Program Cust Comput Mach* (2003), 51–61, https://doi.org/10.1109/FPGA.2003.1227241.

7 Melià-Seguí J, Garcia-Alfaro J & Herrera-Joancomartí J, Multiple-polynomial LFSR based pseudorandom number generator for EPC Gen2 RFID tags, *37th Annu Conf IEEE Ind Electronsoc* (2011) 3820–3825, https://doi.org10.1109/IECON.2011.6119932.

8 Gupta A, Asad A, Meena L & Anand R, IoT and RFID-Based Smart Card System Integrated with Health Care, Electricity, QR and Banking Sectors, *Artif Intell Med Proc Int Symp* (Springer, Singapore) 2023, 253–265.

9 Peris-Lopez P, Hernandez-Castro J C, Estevez-Tapiador J M & Ribagorda A, LAMED — A PRNG for EPC Class-1 Generation-2 RFID specification, *Comput Stand Interfaces*, **31(1)** (2009) 88–97, https://doi.org/10.1016/j.csi.2007.11.013

10 Mandal K, Fan X & Gong G, Warbler: A Lightweight Pseudorandom Number Generator for EPC C1 Gen2 Passive RFID Tags, *Int J RFID Secur* Cryptogr, **2(2)** (2013) 1–10.

11 López A B, Encinas L H, Muñoz A M & Vitini F M, A lightweight pseudorandom number generator for recuring the internet of things*, IEEE Access*, **5** (2017) 27800–27806, https://doi.org/10.1109/ACCESS.2017.2774105.

12 Melià-Seguí J, Garcia-Alfaro J & Herrera-Joancomartí J, J3Gen: A PRNG for Low-Cost Passive RFID, *Sensors*, **13(3)** (2013) 3816–3830, https://doi.org/10.3390/s130303816.

13 Eljadi F M A & ShaikhliI F T A, Dynamic linear feedback shift registers: A review, the *5th Int Conf Inform Commun Technol Muslim World* (IEEE) 2015, 1–5, https://doi.org10.1109/ICT4M.2014.7020598.

14 Che W, Deng H, Tan W & Wang J, A random number generator for application in RFID tags, *Networked RFID Systems and Lightweight Cryptography*, (2008) 279–287, https://doi.org/10.1007/978-3-540-71641-9_16.

15 Melia-Segui J, Garcia-Alfaro J & Herrera-Joancomarti J, A practical implementation attack on weak pseudorandom number generator designs for EPC Gen2 tags, *Wirel Pers Commun*, **59(1)** (2011) 27–42, https://doi.org/10.1007/s11277-010-0187-1

16 Chen J, Miyaj A, Sato H & Su C, Improved Lightweight Pseudo-Random Number Generators for the Low-Cost RFID Tags, *IEEE Trustcom/BigDataSE/ISPA* 2015, 17–24, https://doi.org/10.1109/Trustcom.2015.352.

17 E P C global, EPC radio-frequency identification protocol class-1 generation-2 UHF RFID for communication at 860-960 MHz (2008), https://www.gs1.org/sites/default/files/docs/epc/Gen2_Protocol_Standard.pdf.

18 Feldhofer M & Wolkerstorfer J, Hardware implementation of symmetric algorithms for RFID security, in *RFID Security* (Springer, Boston, MA) 2009, 373–415.

19 Cole P H & Ranasinghe D C, *Networked RFID Systems and Lightweight Cryptography* (Springer) **10** (2008) 157–167.

20 Etrog J Robshaw M & Savry O, *The Possibilities and Limitations of Cryptography in Constrained Devices* (INRIA and Orange Labs) 2009

21 Boni A & Facen A, Ultra low-voltage analog circuits for UHF RFID devices in 180 nm CMOS technology, *Analog Integr Circuits Signal Process*, **63(3)** (2010) 359–367.