A review of cryptographically secure PRNGs in constrained devices for the IoT

A.B. Orúe*, L. Hernández-Encinas, V. Fernández, and F. Montoya

Institute of Physical and Information Technologies (ITEFI)
Spanish National Research Council (CSIC), Madrid, Spain
{amalia.orue,luis,veronica.fernandez,fausto}@iec.csic.es

Abstract. In this work we show a deep review of lightweight random and pseudorandom number generators designed for constrained-devices such as wireless sensor networks and RFID tags along with a study of Trifork pseudorandom number generator for constrained devices.

Keywords: Internet of things, lightweight cryptography, pseudorandom number generator, constrained devices.

1 Introduction

In recent years, tiny computing devices have been designed for many applications in the Internet of Things (IoT) environments, such as radio frequency identification (RFID) tags, smart cards and wireless sensor networks (WSNs), generally referred as low-cost smart devices. Securing this kind of devices is a challenging task due to its very limited power consumption, constrained memory and computing capability. Consequently, a lightweight cryptographic primitives such as block ciphers, stream ciphers, and hash functions must be developed.

True Random (TRNG) and pseudorandom number generators (PRNG) are essential primitives of cryptosystems, used to generate confidential keys, challenges, nonces and in authentication protocols. In constrained devices, cryptographically secure PRNGs are difficult to achieve due to hardware/software limitations. In comparison with other secret key primitives, not many PRNG proposals as such have been found in the literature. Nevertheless they are intrinsically present in many other buildings blocks of other cryptographic primitives. In order to provide compact secure algorithms and protocols that fit in these resource-constrained environments, it is very important to know and understand these constraints first.

The term *resource-constrained environment* is used in describing a development platform that has very little amount of designing space. For example, for hardware implementations, chip area, energy consumption (e.g. battery life), hardware memory, computation latency, and communication bandwidth should be considered to evaluate the lightweight properties. In the case of software implementations, attention to the execution time, the RAM consumption (RAM footprint) and the code size should be paid.

* Corresponding author

In this work, we present a comprehensive review of the lightweight random and pseudorandom number generators designed for constrained-devices such as wireless sensor networks and RFID tags and a study of Trifork, as a suitable and special case of PRNG for constrained devices.

2 Literature survey

In this section, we survey various pseudorandom and true random number generators suitable in some extent to constrained devices.

The generation of random numbers in constrained devices can be divided in two main categories. The first are designs based on TRNGs, where a physical characteristic of the device is used. The second includes the works based on a PRNGs, where a deterministic algorithm is used. A third category uses a combination of the first and the second categories, where the output sequence obtained from a TRNG is used as a seed of a PRNG. Other schemes which produce random numbers by using hash and block cipher algorithms in such a way that overwhelm the limited capabilities of resource-constrained devices, have not been incorporated in this survey.

2.1 PRNG and TRNG target for wireless sensor networks

A *sensor network* is an infrastructure that comprises a group of autonomous, tiny and cheap sensor devices equipped with small battery, low memory and limited processing capability. They are able to sense environmental data such as sound, light and temperature, and communicate with any other sensor device within the same network area and compute the monitored/received data [1]. Due to previously mentioned constraints, standard security primitives can not be directly applied to this area. One of the most common sensor type is TelosB. It has a 16-bit, 8MHz RISC CPU with only 10K RAM, 1024K flash storage and 48K program memory. These restrictions make that the software built for the sensor must also be quite small. Moreover, the *de facto* standard operating system for wireless sensors named TinyOS, has a total code space of approximately 4K. Accordingly, the code size for the all security algorithms must fit within this size limit. Sensor nodes are battery operated and they have extremely limited energy supply. Hence, the implementation of a cryptographic primitive or protocol within a sensor node must also consider the energy impact on its battery life. Many security protocols have been designed to provide security services such as authentication, confidentiality and integrity. Only a few PRNGs have been proposed as such, but they are intrinsically integrated in the majority of the security protocols.

In [2], the authors have developed a low power 8-bit PRNG based on a "free-running timer" for ultra low power sensor network i-Bean that could be used in a general low-power embedded networks. The authors pinpoint the requirements needed for its design and then they proposed a PRNG whose sequence is obtained by XORing the current value of the timer with a key. Then, the timer

values and the key are updated in such a way that one's complement of the timer value becomes the new key and the one's complement of the new random number becomes the next timer value. Furthermore, after the generation of K consecutive random numbers, a re-keying process is done using the checksum-Cyclic Redundancy Check (CRC) of the transmitted/received packets to re-key the generator. The implementation of the generator was done on Microchip PIC 18F8720 platform running at a clock speed of 8 MHz, where the authors obtained a 50 000 random bytes per minute. The sequences of the PRNG were tested using ENT statistical test [3]. This low power PRNG was cryptanalyzed in [4]. The authors observed important security problems of the system; firstly they found some points that are undefined in the target paper such as the exact entropy introduced by the transmitted and received messages in the network and how frequently it is necessary to apply the re-keying algorithm. They also found that the CRC of the transmitted packets has a very low entropy, and moreover, these packets are easily modified by an active attacker. In the cryptanalysis the authors have shown that the PRNG has a remarkably short period, in fact after 3 iterations the generator returns to the same internal values. Additionally the authors demonstrated that the internal secret state of the generator can be easily recovered by eavesdropping only two consecutive outputs.

TinyRNG, a cryptographic pseudorandom number generator (CPRNG) tailored for wireless sensor nodes which uses the transmission of bit errors on this network as a randomness source, was presented in [5]. They have pointed out that these bit errors are difficult to observe and manipulate by an attacker. The erroneous bytes received by a node are added into a cryptographic entropy accumulator. This accumulator is designed using a cipher block chaining message authentication code (CBC-MAC) technique for minimizing the memory requirement of the system by using the same block cipher as the CPRNG. When sufficient entropy is accumulated, the accumulator re-seeds the key of the CPRNG. The CPRNG is indeed a block cipher working in counter mode (acting like a stream cipher) encrypting a counter using the key supplied at programming time. The key also gets updated with the value generated by the accumulator.

In [6] was proposed a TRNG composed of two modules, in one of them a local node generates true random bits based on its sensory measurements derived from a physical process like temperature and humidity, etc., and the other processes the received sequence of true random bits for improving its statistical behaviour. In order to remove redundant and biased information a de-skewing technique is applied. The results thus obtained are passed to a MAC algorithm, for increasing diffusion and confusion over the sequence. The system allows users to select one out of two different MAC algorithms based on hash functions and block ciphers, respectively: a keyed-HMAC with either SHA-512 or MD5 and a Cipher-based Message Authentication Code (CMAC), choosing between DES, 3DES, and AES algorithms. Therefore, every node can produce random data upon request, avoiding the store and transmission of sensitive information, thus reducing vulnerabilities in the network. A proof of concept was done using a WSN formed by two sensors TelosB nodes the first one is used for sampling

sensory data and forward this data to the second one, which acts as Base Station. The Base Station then processes the received data to obtain the final true random sequence, whose good statistical behaviour was confirmed by the NIST statistical suit. The security problems of this TRNG was analyzed in [7]. Authors pointed out a vulnerability of the system due to the complete locality of the process: if a node merely relied on its own measurements, the TRNG could be attacked by violating the analog-digital converter of a node and learning the measurements. To overcome these problems they propose to assign two types of roles to the nodes, a node plays the role of *measurements requesters* and other nodes act as *measurements generator*, which improves the overall robustness of the system since the attacker would have to guess which node are the measurements generators. Hence, they proposed an algorithm that involves two phases: a *k-neighbourhood identification phase* where a tree rooted at the requester node is constructed with depth k, so that exactly one path exists between the requester and any other node; and a *leather messages convergecast phase* where network nodes filter out potential leaders, until a unique leader will be elected at root node. As a proof of concept, the authors have considered a real set of 60 TelosB nodes deployed within an area of $25 \times 15\ m^2$. They conclude that their algorithm is more suitable and efficient for the purpose of Secure Random Number Generation than other methods. The robustness of the TRNG algorithm against several security attacks was also presented.

In [8], a Warbler family as a new PRNG family for low cost smart devices was presented. It is based on the combination of modified *de Bruijn* blocks –which is a generator that combines m primitive nonlinear feedback shift registers of different lengths– together with a nonlinear feedback Welch-Gong (WG) shift registers based in [10]. The authors presented two instances of the Warbler family with different security levels. The first one is suitable for EPC Gen2 RFID tags and was detailed in [11], where it provides the 16-bit random numbers required in the EPC Gen2 standard. The proposed PRNG can be implemented on low-cost Xilinx Spartan-3 FPGA devices with 46 slices. The second one is a security-enhanced version suitable for RFID tags and WSNs. The authors assured that this generator is also resistant to cryptanalytic attacks such as algebraic attacks, cube attacks, time-memory-data tradeoff attacks, etc. The security analysis shows that both proposed instances pass the cryptographic statistical tests recommended by the EPC C1 Gen2 standard and NIST. Their ASIC implementations using a $65nm$ CMOS process demonstrate that the proposed two lightweight instances of the Warbler family can achieve good performance in terms of speed and area and provide ideal solutions for securing low-cost smart devices. However, recently in [25] a distinguish attack to a WG family of stream ciphers shows that every member of this family is vulnerable to linear attacks, which could represent an important threat to the security of Warbler PRNGs.

2.2 PRNG in RFID

RFID is a technology widely used to perform automatic and unique identification of objects. It is used in real world applications that include access control and

supply chain management. A typical RFID system includes tags or transponders, readers or interrogators, and a back-end database. Each RFID tag has a unique identifier, the Electronic Product Code (EPC), which follows the standard EPCglobal Class 1 Generation 2 (EPC Gen2) [9] for passive low-cost RFID systems. This standard establish a platform for RFID protocol interoperability and supports basic security services given by an on-chip 16-bit PRNG and a 16-bit CRC. Therefore, the PRNG is the only component used for achieving the security of the system. Since its publication, there have been many attempts to improve the security of the EPC Gen2 protocols using the CRC, or implementing some schemes based on the passwords defined in the standard. It is worth mentioning that the EPC Gen2 and ISO/IEC 18000-6C standard states that the RFID tags can devote up to 4000 gate equivalent (GE)[1] to security functions.

In [12], the authors propose an oscillator-based TRNG combined with a 16-bit LFSR PRNG, suitable for RFID tags. The TRNG uses as a source of randomness the thermal noise of an analog circuit. Each cycle of the LFSR is modified by XORing the first cell of the LFSR with the output of the oscillator-based TRNG. As a result, a 16-bit random number in 16 clock cycles is obtained. The authors claim that the addition of only a truly random bit, as a random number seed, makes the LFSR output sequence unpredictable and unrepeatable. Finally, they presented two optimization methods to reduce the power consumption of the TRNG. This scheme has been cryptanalyzed in [13], taking advantage of the inherent linearity of the LFSR. They showed that with only very few observations, feedback polynomial of the LFSR can be successfully retrieved. Additionally the authors propose a similar and improved PRNG scheme, that relies on a multiple primitive polynomial LFSR core perturbed by a TRNG. Such design use eight primitive polynomials, a TRNG, and a decoding circuit whose inputs come from the true random source. The decoding circuit was designed avoiding the use of the same primitive polynomial consecutively. To generate a pseudorandom bit, one of the primitive polynomials is chosen at each clock cycle according to the decoding logic and a true random bits. The resulted PRNG generates a 16-bit random number in 16 clock cycles and the security of the PRNG relies on the TRNG in the same manner as in the former design. Finally a hardware complexity estimation has been presented.

The LAMED PRNG target for RFID tags applications which uses 32-bit keys and a pre-established initial states was presented in [14]. The algorithm consists of simple operations such as bit rotations, modular algebra and bitwise operations. The design uses a Mersenne Twister Generator(MTG) to provide input to the different operations of a Genetic Programming Algorithm (GPA). The authors propose two versions of the algorithm: One is a 32-bit PRNG and the other is a 16-bit PRNG for EPC Gen2 compatibility. To reduce the output length from 32 to 16 bits, the 32-bit output is divided in two halves and XORed to obtain the 16-bit output sequence. The compliance to EPC G2 security requirements

[1] One GE is equal to the area needed by two-input NAND gate with the lowest driving strength of the corresponding technology. Hence, the area in GE is obtained by dividing the area in μm^2 by the area of a two-input NAND gate.

was successfully achieved. The NIST, ENT and Diehard statistical suit tests were used validate the randomness of the generators. Finally the authors analyzed the hardware complexity of the generators which meets the requirements imposed by low-cost RFID tags technology, i.e., the gate count is below 2K gates.

In [15], authors have explored different architectures to perform the modular multiplications in the implementation of the BBS PRNG. They found that the major problems in the BBS implementation is the excessive needed circuit area and computation time; specifically those related to the use of modular multiplications. The security of the BBS generator depends on the factorization intractability of the integer module. Moreover, they suggest that an equivalent security to a key length of 2^{32} bits and 2^{64} bit may be enough for RFID applications, which is equivalent to a BBS module of 160 and 512 bits, respectively. Therefore, they implemented two BBS schemes. To determine the best of the two schemes the total area of the circuit was considered. Considering this, Montgomery iterative architecture was the most efficient and hence, the authors recommended it for constrained devices like RFID tags or sensor networks nodes. In the implementation, they used a Spartan-3 FPGA device, specifically the XC3S1500L (4-input LUTs). It is worth to mention that NIST recommendations for years 2016-2030, a module from $n = 3072$ to $n = 15360$ bits length, equivalent to a security level of 2^{128} to 2^{256} respectively, must be used for cryptographic applications. Therefore, this PRNG has a low security level.

A new generator for RFID tags that combines a physical source of true randomness and a deterministic LFSR, was proposed in [16]. The proposal has four main blocks: an LFSR configured with a multiple-polynomial tap architecture, a Decoding Logic, a Polynomial Selector and a TRNG module based on the oscillator-based TRNG designed in [12]. Basically the output of the TRNG is fed in the Decoding Logic that manages the Polynomial Selector. The feedback polynomials are implemented as a wheel, which rotates depending on the bit value given by the TRNG module, which prevents the natural linearity of any LFSR. The authors provide the approximated hardware complexity of the PRNG, sized in GE, to measure its suitability for EPC Gen2 requirements.

In [17] a PRNG named MeTuLR: Mersenne Twister generator (MTG) for RFID tags was presented. It is a scheme for 16-bit architecture to obtain 32-bit PRNs that uses extracted TRNs from hardware as input for the MTG. To eliminate the bias of the TRN they use an ultra-light scheme, hash equivalent, that consists of a bitwise XOR, AND and rotation operations that consumes less area and clock cycles. The scheme substituted the shift operations with rotations; matrix iterations of the original MTG are eliminated and compensated by means of rotations. The final TRNs have passed the randomness test.

A modification of a previous PRNG [16] was proposed in [18], which was named J3Gen. It meets the requirements specified for the PRNG in the EPC Gen2 standard. The design combines a thermal-noise TRNG and a dynamic-LFSR (DLFSR) of n cells. A DLFSR is just an LFSR whereby the feedback polynomial changes dynamically. The proposal has four main blocks: a TRNG module based on the oscillator-based TRNG by [12], a DLFSR architecture, a

Decoding Logic and a Polynomial Selector. The output of the TRNG is fed to the Decoding Logic that manages the Polynomial Selector. The feedback polynomials are implemented as a wheel, i.e., they rotate depending on the bit value given by the TRNG module, the rotations are performed every l cycles. This prevents the natural linearity of any LFSR. The authors measure the suitability of the design for EPC Gen2 providing the approximated hardware complexity of PRNG, sized in GE. The proposed architecture results in a security equivalent-key size of 372 bits. The nonlinearity of the design and its suitability regarding the standards was validated, and the randomness requirements was tested through a statistical analysis. The power consumption was evaluated by means of SPICE simulation.

This design has been successfully cryptanalyzed by Peinado et al. [19] who showed that the algorithm is insecure by means of two attacks; a probabilistic attack and a deterministic attack. The probabilistic attack was based on solving linear equation systems, which allows to recover the set of feedback polynomials, which constitute the secret information of PRNG. The deterministic attack was based on a decimation of the output sequence. It was showed by the authors that the entire output sequence of PRNG can be reconstructed by the knowledge of only a few bits of it. The authors also presented some security recommendations to improve the J3Gen PRNG. They suggested that including non-primitive polynomials in the pool of feedback polynomials could prevent the PRNG from a brute force attack, only if it is done carefully, since non-primitive polynomials produce sequences whose statistical properties are not guaranteed. They strongly recommend not to use the rotation cycle $l = 1$.

In [20] a low-area implementation of Warbler PRNG firstly presented in [11,8]. The implementation was done in CMOS $65nm$ and CMOS $130nm$ ASICs, providing the area, maximum clock frequency, and total power consumption results. The proposal includes two design options for the counter: a binary counter and a LFSR counter. The LFSR counter-based design resulted better than the binary counter-based one in terms of area and power consumption. Recently in [25] a distinguish attack to WG family of stream ciphers, shows that every member of this family, used in the Warbler design, is vulnerable to linear attacks, which could represent a threat to the security of Warbler PRNGs.

It can be seen that the great majority of the TRNG and PRNG proposals for constrained-device environments have been successfully attacked either by taking advantage of the CRC function or the LFSR linearity like in [4,13,19,21,22]. As these technologies move toward a great ubiquity, they will become more and more vulnerable to be attacked, so advanced new techniques are required to improve its security, performance and reliability.

3 Analysis of Trifork suitability for using in constrained devices

Trifork is a cryptographically secure family of PRNG based on a combination of three coupled Perturbed Lagged Fibonacci Generators (PLFG) [23]. Two PLFG are combined by the bitwise XOR addition, to form the output of the joint

generator; the third one will remain completely hidden. In this way the analysis of the output sequence is useless for determining the system parameters, system state nor keys. The three PLFG are interconnected and perturbs each other in a cyclic fashion, by means of the XOR addition of the left-shifted output of the precedent PLFG, being the left-shift of about $\frac{N}{2}$ bits, were N the bit word size. Its period is much longer than the conventional Lagged Fibonacci generator. Its output is unpredictable and the generated sequence passes successfully the most stringent randomness test suites. Implementation in hardware and software has been analyzed. The cost in a hardware implementation (ASIC) is approximately equivalent to the area in silicon. Usually the hardware area is measured in GE, which allows an estimation technically neutral of the physical space required by a design. We have based our estimation of the GE according to [24].

An implementation specially adapted for RFID 16 bit random number generation, will consist of three PLFG with lengths of 3, 3 and 2 words respectively, and $N = 16$ bit size. Each PLFG is perturbed by a XOR addition of 7, 8 and 9 bit left-shift of the output of the precedent one. The resulting key length is 128 bits and the state space is $2^{128} - 1$. Tables 1 and 2 illustrate the total GE needed for its implementation with 8 and 16 bits word size. It can be seen that the required amount of GE is considerably less than the maximum allowed 4K GE. The performance in a software implementation over a low power 8 bit ATmega328P microcontroller was about 22.2 cycles per bit.

Table 1: GE with 8 bit word size.

Operator	Used	GE–UMC 90nm	GE–AMIS 0.35μm
XOR	20	49.60	46.60
ADD	24	178.80	152.16
Register	128	584.96	853.76
Control	20%	162.67	210.50
Total GE		976.03	1263.02

Table 2: GE with 16 bit word size.

Operator	Used	GE–UMC 90nm	GE–AMIS 0.35μm
XOR	40	99.20	93.20
ADD	48	357.60	304.32
Register	128	584.96	853.76
Control	20%	208.35	250.25
Total GE		1250.11	1501.53

4 Conclusions

In this work we present a comprehensive literature review of the TRNG and PRNG designed for constrained-devices commonly deployed in the IoT. We have summarized the features of each PRNGs and the best cryptanalysis results. In Table 3 we show the algorithms analyzed and the limitations pointed out by their cryptanalysis, ordered from the best to the worst proposal presented. We have found several PRNG families vulnerable to attacks, or not recommended for using in new applications and, thus, should be avoided. With reference to the most recent PRNG designs, a cryptanalytic work must be done to verify their security before they can be promoted as secure primitives. Finally we have studied the feasibility of using Trifork PRNG for constrained devices, and verified that its design in GE fits to the recommended values. However, a further study of other characteristics required by the standard is necessary to be considered as a candidates for the IoT.

Table 3: PRNG for Wireless sensor networks and for RFID.

Proposals	**Limitations**
[8,5]	None found.
[2]	Very low entropy of the CRC of the transmitted packets [4]
[6]	Complete locality of the process of generation of the TRN, a local node relied on its own measurements; therefore, the TRNG could be attacked [7]
[11,20,14,17]	None found.
[12]	The linearity of the LFSR permit to retrieve the feedback polynomial of the LFSR [13]
[15]	Very low key space: $2^{64} \ll 2^{128}$ bits recommended by NIST
[16,18]	The linearity of the system [19]

Acknowledgments. This work has been supported by the European Union FEDER funds distributed through Ministerio de Economía y Competitividad (Spain) under the project TIN2014-55325-C2-1-R (ProCriCiS), and Comunidad de Madrid (Spain) under the project S2013/ICE-3095-CM (CIBERDINE).

References

1. Conti, M. Secure Wireless Sensor Networks. Threats and solutions, volume 65, Advances in Information Security. Springer-Verlag New York, (2015)
2. Seetharam, D., Rhee, S. An efficient pseudo random number generator for low-power sensor networks. 29th Annual IEEE International Conference on Local Computer Networks, 560–562, (2004)
3. Walker, J. ENT a pseudorandom number sequence test program. `https://www.fourmilab.ch/random/`
4. Peris-López, P., Hernández-Castro, J.C., Estévez-Tapiador, J.M., San Millán, E., van der Lubbe, J.C.A. Security flaws in an efficient pseudo-random number generator for low-power environments, SEWCN, 25–35, (2009)
5. Francillon, A., Castelluccia, C. TinyRNG: A cryptographic random number generator for wireless sensors network nodes. IEEE 2007 5th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, 1–7, (2007)
6. Gaglio, V., De Paola, A., Ortolani, M., Lo Re, G. A TRNG exploiting multi-source physical data. Proceedings of the 6th ACM Workshop on QoS and Security for Wireless and Mobile Networks, Q2SWinet'10, 82–89, New York, USA, (2010)
7. Lo Re, G., Milazzo, E., Ortolani, M. Secure random number generation in wireless sensor networks. Proceedings of the 4th International Conference on Security of Information and Networks (SIN'11). ACM, 175–182, (2011)
8. Mandal, K., Fan, X., Gong, G. Design and implementation of Warbler family of lightweight pseudorandom number generators for smart devices. ACM Transactions on Embedded Computing Systems, 15, 1–28, (2016)
9. EPCglobal. EPC radio-frequency identity protocols generation-2 UHF RFID, specification for RFID air interface, protocol for communications at 860 MHz–960 MHz. Version 2.0.1 Ratified, (2015)
10. Nawaz, Y., Gong, G. WG: A family of stream ciphers with designed randomness properties. Information Sciences 178(7), 1903–1916, (2008)

11. Mandal, K., Fan, X., Gong, G. Warbler: A lightweight pseudorandom number generator for EPC C1 Gen2 passive RFID tags. International Journal of RFID Security and Cryptography 2, 1–4, (2013)
12. Che, W., Deng, H., Tan, X., Wang, J. A random number generator for application in RFID tags. Networked RFID Systems and Lightweight Cryptography: Raising Barriers to Product Counterfeiting, 279–287, (2008)
13. Melià-Seguí, J., Garcia-Alfaro, J., Herrera-Joancomarti, J. Analysis and Improvement of a Pseudorandom Number Generator for EPC Gen2 Tags, 34–46, (2010)
14. Peris-López, P., Hernández-Castro, J.C, Estévez-Tapiador, J.M., Ribagorda, A. LAMED–a PRNG for EPC Class-1 Generation-2 RFID specification. Computer Standards and Interfaces 31(1), 88–97, (2009)
15. Peris-López, P., San Millán, van der Lubbe, J.C.A., Entrena, L.A. Cryptographically secure pseudo-random bit generator for RFID tags. International Conference for Internet Technology and Secured Transactions (ICITST), 1–6, (2010)
16. Melià-Seguí, J., Garcia-Alfaro, J., Herrera-Joancomartí, J. Multiple-polynomial LFSR based pseudorandom number generator for EPC Gen2 RFID tags. 37th Annual Conference of the IEEE Industrial Electronics Society, 3820–3825, (2011).
17. Özcanhan, M. H., Dalkiliç, G., Gürle, M.C. An ultra-light PRNG for RFID tags. Lent R. Gelenbe E. (Ed), Computer and Information Sciences III, 231–238. Springer, (2013)
18. Melià-Seguí, J., Garcia-Alfaro, J. Herrera-Joancomartí, J. J3Gen: A PRNG for low-cost passive RFID. Sensors 13, 3816–3830, (2013)
19. Peinado, A., Munilla, J., Fúster-Sabater, A. EPC Gen2 pseudorandom number generators: Analysis of J3Gen. Sensors 14(4), 6500–6515, (2014)
20. Yang, G., Aagaard, M.D., Gong, G. Efficient hardware implementations of the Warbler pseudorandom number generator. IACR ePrint Archive, 2015:789, (2015)
21. Sohrabi-Bonab, Z., Alagheband, M. R., Aref, M.R. Formal cryptanalysis of a CRC-based RFID authentication protocol. 2014 22nd Iranian Conference on Electrical Engineering (ICEE), 1642–1647, (2014)
22. Safkhani, M., Bagheri, N. For an EPC-C1 G2 RFID compliant protocol, CRC with concatenation: No; PRNG with concatenation: Yes. Cryptology ePrint Archive, 2013:490, (2013)
23. Orúe, A.B., Hernández-Encinas, L., Montoya, F. Trifork, a new Pseudorandom Number Generator Based on Lagged Fibonacci Maps. Journal of Computer Science and Engineering 2(2), 46–51 (2010)
24. Martín, H., Peris-López, P., Tapiador, J. E., San Millán, E. An estimator for the ASIC footprint area of lightweight cryptographic algorithms. IEEE Transactions on Industrial Informatics, 10(2), 1216–1225, (2014)
25. Mabin, J., Gautham, S., Balasubramanian, R. Distinguishing Attacks on (Ultra) Lightweight WG Ciphers. Lightweight Cryptography for Security and Privacy: 5th International Workshop, LightSec 2016, Turkey, 45–59, 2017.