Multiple-Polynomial LFSR based Pseudorandom Number Generator for EPC Gen2 RFID Tags

Joan Melià-Seguí
Universitat Oberta de Catalunya
Roc Boronat 117
08018, Barcelona - Spain
Email: melia@uoc.edu

Joaquin Garcia-Alfaro
Institut Telecom, Telecom Bretagne
02, rue de la Chatagneraie
Cesson-Sevigne 35576 - France
Email: joaquin.garcia-alfaro@acm.org

Jordi Herrera-Joancomartí
Universitat Autònoma de Barcelona
Edifici Q, Campus de Bellaterra
08193, Bellaterra - Spain
Email: jherrera@deic.uab.es

***Abstract*—We present a lightweight pseudorandom number generator (PRNG) design for EPC Gen2 RFID tags. It is based on a linear feedback shift register (LFSR) configured with multiple feedback polynomials that are selected by a physical source of randomness. The proposal successfully handles the inherent linearity of LFSR based PRNGs and satisfies the statistical requirements imposed by the EPC Gen2 standard. Statistical analysis of the sequences generated by our generator confirms the validity of the proposed technique. We show that our proposal has, moreover, a simpler hardware implementation and energy consumption than previous designs reported in the literature.**

I. Introduction

The Electronic Product Code (EPC) is a Radio Frequency IDentification (RFID) technology for the automatic identification of objects. Its main interface is the EPC Class 1 Generation 2 UHF Air Interface Protocol standard (EPC Gen2 for short) [1]. EPC Gen2 compliant RFID tags are passive electronic labels without self-power supply. They are energized by the electromagnetic field of RFID readers. Distances from which Gen2 labels can be energized and interrogated by the readers is, typically, up to five meters.

The integration of security features on-board of EPC Gen2 tags faces several challenging constraints such as cost, compatibility regulations, power consumption, and performance requirements. EPC Gen2 tags only consider two main security elements: a 16-bit pseudorandom number generator (PRNG) and password-protected operations. The pseudorandomness offered by the on-board PRNG is, indeed, used to protect the password-protected operations. The PRNG is also used as an anti-collision mechanism for inventorying processes [1]; and to acknowledge other EPC Gen2 specific operations (e.g., memory writing, decommission of tags, and self-destruction). The on-board 16-bit PRNG is, therefore, the crucial component that guarantees the security of a Gen2 tag.

As required by the EPC specification, every commercial Gen2 tag does implement an on-board 16-bit PRNG. However, manufacturers are reluctant to provide their designs [2]. They simply refer to testbeds that show the accomplishment of some compatibility requirements. They fail to offer convincing information about the security of their designs. This is mostly security through obscurity, which is always ineffective in security engineering, as it has been shown with the disclosure of the PRNG used in the MIFARE Classic chip [3] that has shown a vulnerable PRNG. The use of weak PRNGs that allow the predictability of the outgoing sequences do also introduce important security flaws in EPC Gen2 communications. For example, it might allow an adversary to bypass the security of the password-protected commands defined in the Gen2 standard (e.g., the *access* and the *kill* commands [1]).

In this paper, we propose a lightweight PRNG scheme for EPC Gen2 tags. We present an attractive approach based on the use of linear feedback shift registers (LFSRs). The simplicity of using LFSRs and their low hardware complexity, while still providing efficient statistical properties, guarantees the ease and efficient adaptation of our scheme to meet the requirements of the EPC Gen2 specifications. Moreover, our solution handles the inherent linearity of LFSR designs in order to satisfy the security randomness requirements imposed by the EPC Gen2 specification. An evaluation of the hardware complexity and power consumption of our proposal confirms, moreover, that our design has a simpler implementation than any other previous schemes reported in the related literature.

Paper Organization — Section II describes of our proposed PRNG. Section III presents an example of its internal execution. Section IV evaluates statistical properties, hardware complexity and power consumption of our PRNG. Section V surveys related work. Section VI closes the paper.

II. Our PRNG Proposal

The main challenge to obtain an efficient PRNG is how to guarantee the generation of sequences with (almost) true random properties, while also addressing efficiency and computational complexity. Indeed, the low power, chip area and output rate (among other constraints) of EPC Gen2 tags makes very difficult the use of true random number generator (TRNG) designs based on, e.g., thermal noise, high frequency sampling or fingerprinting, for a real-time generation of full length pseudorandom sequences. We propose to address this problem by combining a physical source of true randomness and a deterministic LFSR.

This work has been supported by the Spanish Ministry of Science and Innovation, the FEDER funds under the grants TSI2007-65406-C03-03 E-AEGIS, CONSOLIDER-INGENIO CSD2007-00004 ARES, TIN2010-15764 N-KHRONOUS.

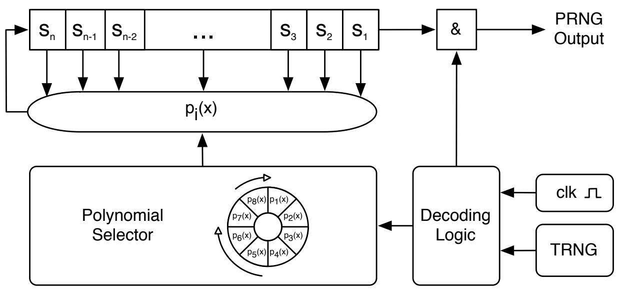

Fig. 1. Block diagram of our PRNG proposal

Figure 1 depicts a block diagram of our proposal. It gets inspiration from a dynamic LFSR-based testing selection scheme presented by Hellebrand *et al.* in [4], [5]. Indeed, it substitutes the static feedback polynomial configuration of an LFSR by a multiple feedback primitive polynomials configuration architecture. The different feedback primitive polynomials are connected to the LFSR by a *Decoding Matrix Unit* that selects each single feedback polynomial. After a given number of LFSR cycles, the *Polynomial Selector* shifts its position towards a new configuration. The number of shifts, i.e., the corresponding selection of each primitive polynomial at a certain LFSR cycle, is determined by a true random bit (hereinafter denoted as trn) that is obtained from a physical source of randomness. More details are presented in the sequel.

A. Linear Feedback Shift Register

Our proposal relies on a n-cell LFSR core perturbed by a physical source of randomness. LFSRs produce pseudorandom sequences with good statistical values. They are very fast and efficient in hardware implementations, and quite simple in terms of computational requirements [7]. This makes the use of LFSRs an ideal system for both energy and computational constrained environments. Moreover, LFSRs follow the same hardware scheme as those cyclic redundancy check (CRC) functions already included in the EPC Gen2 standard [1]. Therefore, current EPC Gen2 tags shall be able to execute LFSR-based functions in the same hardware.

B. Polynomial Selector

The Polynomial Selector is the responsible of the linearity avoidance in our scheme. A set of m primitive feedback polynomials is selected, and each single feedback polynomial is used depending on the value of the truly random bit provided by the TRNG module. The feedback polynomials are implemented as a *wheel*, which rotates depending on the bit value given by the TRNG module. If the truly random bit is a logical 0, the *wheel* rotates one position, that is, it selects the next feedback polynomial. Instead, if the truly random bit is a logical 1, then the *wheel* rotates two positions, that is, the Polynomial Selector jumps one feedback polynomial and selects the next one.

C. Decoding Logic

The Decoding Logic is the responsible of managing the internal PRNG clock. It activates and deactivates the PRNG modules for its proper performance. The internal PRNG modules have different activation and deactivation timings. Depending on the internal clock frequency, f_{clk}, some modules such as the LFSR or the TRNG need different activation cycles. For example, the trn sampling in the TRNG module is activated only once for each PRNG output.

The Decoding Logic also manages the trn obtained from the TRNG module, rotating the Polynomial Selector with regard to the trn value. This action is performed using l cycles, lower than the n cycles needed in the LFSR, to avoid pseudorandom sequences generated from a single feedback polynomial.

D. Thermal-noise TRNG

Regarding the physical source of randomness (trn), there are different proposals to derive true random sequences of bits from the hardware of a radio-frequency identification (RFID) tag. The technique that we use in our design is an oscillator-based high frequency sampler by Che *et al.* [8], that offers high simplicity and suitability for EPC Gen2 designs. The output of the TRNG is fed to the Decoding Logic which, in turn, manages the Polynomial Selector.

III. Sample PRNG Execution

After the description of the logic components of the system, we show an example of the proposed PRNG internal execution. The goal of this section is to provide better comprehension of the PRNG performance details.

The specific parameters needed in each internal module are defined in Table I, while Table II specifies the chosen feedback polynomials. The LFSR size n has been fixed to 16 and the total number of different feedback polynomials m has been set to 8. Such values ensure an appropriate trade-off between pseudorandomness and hardware complexity, as we discuss in Section IV. We take as the initial LFSR state the value $v_0 = 0x1$ (which represents a logical 1 in the less significant bit, in hexadecimal notation).

TABLE I
Design parameters summary

Size of LFSR (bits)	$n = 16$
Number of feedback polynomials on tag	$m = 8$
Internal clock frequency	$f_{\mathrm{clk}} = 100\,\mathrm{kHz}$
trn sampling period	$f_r = 16$
Polynomial Selector update period	$l = 15$

Table III details each LFSR state for 32 shift cycles (rows) providing 32 outputted PRNG bits (column *Tx*) consisting in two 16-bit sequences. Since the trn sampling frequency is $f_r = 16$ cycles, this 32 shift cycles need two true random values, that in the example have been set to $r_1 = 0$ and $r_2 = 1$.

The system starts with $p(x)_1$ and outputs $l = 15$ bits until the TRNG module transfers a bit with value $r_0 = 0$ to the

TABLE II
FEEDBACK POLYNOMIALS ($n = 16$)

Primitive polynomials
$p_1(x) : 1 + x + x^5 + x^6 + x^7 + x^{11} + x^{16}$
$p_2(x) : 1 + x^4 + x^5 + x^6 + x^7 + x^{11} + x^{16}$
$p_3(x) : 1 + x + x^3 + x^4 + x^5 + x^6 + x^7 + x^{11} + x^{16}$
$p_4(x) : 1 + x^3 + x^5 + x^6 + x^{10} + x^{11} + x^{16}$
$p_5(x) : 1 + x^5 + x^6 + x^{11} + x^{16}$
$p_6(x) : 1 + x^5 + x^6 + x^{10} + x^{11} + x^{13} + x^{16}$
$p_7(x) : 1 + x^4 + x^5 + x^6 + x^{10} + x^{11} + x^{16}$
$p_8(x) : 1 + x + x^3 + x^4 + x^5 + x^6 + x^{10} + x^{11} + x^{16}$

TABLE III
LFSR ITERATION EXAMPLE

	S_{16}	S_{15}	S_{14}	S_{13}	S_{12}	S_{11}	S_{10}	S_9	S_8	S_7	S_6	S_5	S_4	S_3	S_2	S_1	**Tx**
v_0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
Coeff.	x^1	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	x^{10}	x^{11}	x^{12}	x^{13}	x^{14}	x^{15}	x^{16}	
$p_1(x)$	1	0	0	0	1	1	1	0	0	0	1	0	0	0	0	1	
1:	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**
2:	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**
3:	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	**0**
4:	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	**0**
5:	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	**0**
6:	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	**0**
7:	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	**0**
8:	1	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	**0**
9:	0	1	0	0	1	1	1	1	1	0	0	0	0	0	0	0	**0**
10:	1	0	1	0	0	1	1	1	1	1	0	0	0	0	0	0	**0**
11:	1	1	0	1	0	0	1	1	1	1	1	0	0	0	0	0	**0**
12:	1	1	1	0	1	0	0	1	1	1	1	1	0	0	0	0	**0**
13:	1	1	1	1	0	1	0	0	1	1	1	1	1	0	0	0	**0**
14:	1	1	1	1	1	0	1	0	0	1	1	1	1	1	0	0	**0**
15:	0	1	1	1	1	1	0	1	0	0	1	1	1	1	1	0	**0**
Coeff.	x^1	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	x^{10}	x^{11}	x^{12}	x^{13}	x^{14}	x^{15}	x^{16}	
$p_2(x)$	0	0	0	1	1	1	1	0	0	0	1	0	0	0	0	1	
16:	0	0	1	1	1	1	1	0	1	0	0	1	1	1	1	1	**0**
17:	1	0	0	1	1	1	1	1	0	1	0	0	1	1	1	1	**1**
18:	1	1	0	0	1	1	1	1	1	0	1	0	0	1	1	1	**1**
19:	1	1	1	0	0	1	1	1	1	1	0	1	0	0	1	1	**1**
20:	1	1	1	1	0	0	1	1	1	1	1	0	1	0	0	1	**1**
21:	0	1	1	1	1	0	0	1	1	1	1	1	0	1	0	0	**1**
22:	1	0	1	1	1	1	0	0	1	1	1	1	1	0	1	0	**0**
23:	0	1	0	1	1	1	1	0	0	1	1	1	1	1	0	1	**0**
24:	0	0	1	0	1	1	1	1	0	0	1	1	1	1	1	0	**1**
25:	0	0	0	1	0	1	1	1	1	0	0	1	1	1	1	1	**0**
26:	0	0	0	0	1	0	1	1	1	1	0	0	1	1	1	1	**1**
27:	1	0	0	0	0	1	0	1	1	1	1	0	0	1	1	1	**1**
28:	1	1	0	0	0	0	1	0	1	1	1	1	0	0	1	1	**1**
29:	1	1	1	0	0	0	0	1	0	1	1	1	1	0	0	1	**1**
30:	0	1	1	1	0	0	0	0	1	0	1	1	1	1	0	0	**1**
Coeff.	x^1	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	x^{10}	x^{11}	x^{12}	x^{13}	x^{14}	x^{15}	x^{16}	
$p_3(x)$	1	0	1	1	1	1	1	0	0	0	1	0	0	0	0	1	
31:	1	0	1	1	1	0	0	0	0	1	0	1	1	1	1	0	**0**
Coeff.	x^1	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	x^{10}	x^{11}	x^{12}	x^{13}	x^{14}	x^{15}	x^{16}	
$p_4(x)$	0	0	1	0	1	1	0	0	0	1	1	0	0	0	0	1	
32:	1	1	0	1	1	1	0	0	0	0	1	0	1	1	1	1	**0**

Decoding Logic module. Then, a consecutive (but different) feedback polynomial is selected in the Polynomial Selector module, that is, $p_2(x)$. This generates the next $l = 15$ LFSR shifts with $p_2(x)$ until the next trn is obtained. The trn value for this PRNG update is $r_1 = 1$, hence, the Decoding Logic rotates the Polynomial Selector one position at shift 31, and another position at shift 32. Then, $p_2(x)$ is used 14 cycles, $p_3(x)$ is used one cycle, and $p_4(x)$ is used one cycle in this PRNG update and 14 cycles in the next PRNG update (not included in Table III).

IV. EPC GEN2 SUITABILITY

We discuss the suitability of our proposal regarding the restrictions that the EPC standard impose. We analyze three important parameters of our scheme: statistical requirements stated by the EPC Gen2 standard for pseudorandom sequence generation, hardware complexity and power consumption.

A. Statistical performance

Detailed in the EPC Gen2 standard [1], the requirements for the pseudorandom sequences generation can be summarized as follows:

1) The probability that any single 16-bit sequence j drawn from the generator shall be bounded by

$$\mathrm{P}_{\min} = \frac{0.8}{2^{16}} < \mathrm{Prob}(j) < \mathrm{P}_{\max} = \frac{1.25}{2^{16}}$$

2) Among a tag population of up to ten thousand tags, the probability that any two tags simultaneously generate the same 16-bit sequence shall be less than 0.1%.
3) The chance of guessing the next 16-bit sequence generated by a tag shall be less than 0.025% even if all previous outputs are known to an adversary.

To confirm the suitability of our proposal for handling the statistical and randomness requirements defined above, we generate ten different sequences using our PRNG. Values presented in this subsection are the ones obtained by using the PRNG parameters defined in Section III. However, similar results have been obtained when varying parameters n, m, l and the feedback polynomials.

To confirm the achievement of the first requirement, we analyze the frequency of occurrence of each sequence generated from our generator. The results confirm (see Figure 2) that, after analyzing 30 million 16-bit sequences, the probability of occurrence of any given value lies between $P_{\min} = \frac{0{,}8}{2^{16}}$ and $P_{\max} = \frac{1{,}2}{2^{16}}$. Furthermore, our proposal achieves similar statistical results with Random.org true random sequences [9], based on its frequency properties.

The second property for building a EPC Gen2 compliant PRNG requires that two simultaneous identical sequences must not appear with more that 0.1% for a population up to 10,000 tags. To test this property, 10,000 instances of our generator, initialized at random, are used to simulate a real population of 10,000 tags. The obtained results, shown in Table IV, verify that, after ten tests of 1,000 iterations each, none of them show a simultaneous identical sequence rate higher than 0.03793%.

Finally, to statistically confirm the fulfillment of the third property, we conducted a series of correlation tests based on

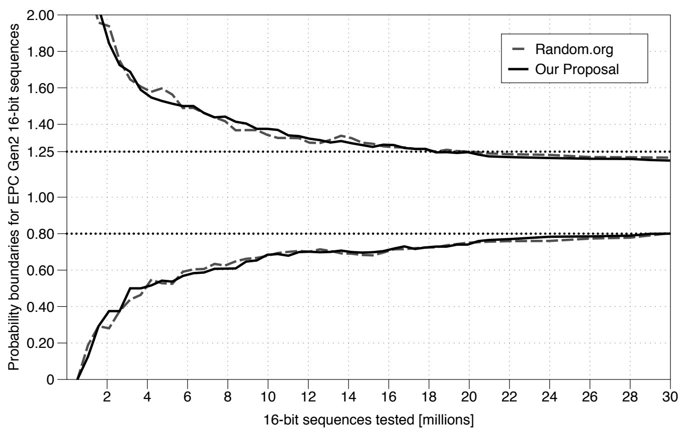

Fig. 2. Compatibility with the $1st$ EPC Gen2 requirement

the T_i sequences. Each test computes the degree of dependence of the ongoing bits regarding their predecessors. We can see by looking the results shown in Table V, that all the tests are under the requested values.

B. Hardware

The goal of this section is to provide the approximated hardware complexity of our PRNG, sized in logic gates equivalence (GE), to measure its suitability for EPC Gen2 requirements. The scientific literature quantifies in between 2,000 and 5,000 equivalent logic gates (GEs) the intended area for security operations in EPC Gen2 ICs [10].

Regarding the system parameters summaryzed in Table I, the ones that significantly impact on the hardware complexity are the size of the LFSR n, and the number of selected primitive polynomials m. Notice, as well, that the implementation of the exact choice of the m polynomials will impact on the hardware complexity.

The Linear Feedback Shift Register module and the Polynomial Selector module have been introduced separately in Section II due to their specific function inside the scheme. However, in the hardware layer design, a different approach shall be used. Although the feedback polynomial is an intrinsic part of the LFSR, we consider its implementation in the Polynomial Selector module, due to the multiple-polynomial configuration. Figure 3 depicts the LFSR module plus the Polynomial Selector, in the specific case of polynomials in Table II. Then, once we fix the cell number of the LFSR and the number of feedback polynomials, we analyze how the selection of different parameters affects to the hardware complexity.

The LFSR size $n = 16$ is selected due to compatibility reasons with the EPC Gen2 technology, since the standard requires the generation of 16-bit pseudorandom nonces [1]. Hence, the polynomials which determine the feedback content are also limited to 16 coefficients. Furthermore, the current EPC Gen2 standard specifies as a mandatory implementation a 16-bit CRC based on LFSR. Thus, we can certainly state that our proposal can be executed in EPC Gen2 labels, since they currently run similar hardware.

Regarding the hardware constrained environment, the on-board implemented polynomials have been fixed to $m = 8$. This value offers good statistical properties (cf. Section IV-A) with a low hardware complexity, since only 3 bits are necessary to manage the selected polynomials. Adding more on-board polynomials do not improve the statistical properties of the system, but on the contrary, it implies an exponential increase on necessary logical gates to build the system. Hence, $m = 8$ offers a suitable trade-off between security and hardware cost, which is the goal of our design.

Regardless of the fixed number of implemented polynomials, the hardware cost of each possible combination may vary. Assuming polynomials of degree 16, we could find 2,048 of them which are primitive. Thus, we can choose a total of $\binom{2,048}{8}$ possible combinations, near 2^{73}. Each Polynomial Selector is implemented based on the coefficients of each combination of polynomials, which is translated into a specific amount of logical components. Based on these parameters, and the necessary hardware to implement each combination of polynomials, we have calculated a worst case amount of 180 necessary GEs to build the set of polynomials, based on two-gates logic such as the ones depicted in Figure 3.

We provide in Table VI the GE counting of the three main modules: the LFSR module, the Polynomial Selector module and the Decoding Logic module. These three elements add up to the most representative amount of GEs. For the LFSR implementation purpose, we use the *D-flip-flop* (DFF) model specified at [11] composed by 18 CMOS transistors. Hence, a D-FF can be measured with approximately 4.5 GE.

The Polynomial Selector module includes the polynomials implementation and the logic hardware to select each polynomial. This hardware complexity is considerably low compared with the 2,000 to 5,000 GE estimated in the literature for security operations [10], thus suitable for our PRNG scheme.

We then provide the physical source of randomness assumed for our generator. For the gate equivalence of this component,

TABLE IV
EPC GEN2 SECOND RANDOMNESS PROPERTY TEST

Test (% rate)	$1st$	$2nd$	$3rd$	$4th$	$5th$
Same 16-bit sequence	0.03777	0.03784	0.03793	0.03772	0.03772
Test	$6th$	$7th$	$8th$	$9th$	$10th$
Same 16-bit sequence	0.03768	0.03757	0.03765	0.03783	0.03749

TABLE V
EPC GEN2 THIRD RANDOMNESS PROPERTY TEST

Sequence	$1st$	$2nd$	$3rd$	$4th$	$5th$
Correlation	0.000001	0.000057	-0.000088	-0.000073	-0.000073
Sequence	$6th$	$7th$	$8th$	$9th$	$10th$
Correlation	-0.000038	0.000134	0.000082	-0.000097	-0.000001

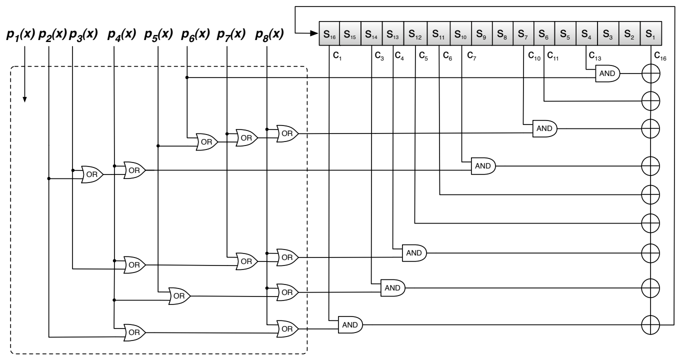
$p_1(x)$ $p_2(x)$ $p_3(x)$ $p_4(x)$ $p_5(x)$ $p_6(x)$ $p_7(x)$ $p_8(x)$
S_{16} S_{15} S_{14} S_{13} S_{12} S_{11} S_{10} S_9 S_8 S_7 S_6 S_5 S_4 S_3 S_2 S_1
C_1 C_3 C_4 C_5 C_6 C_7 C_{10} C_{11} C_{13} C_{16}
AND
OR

Fig. 3. Multiple Polynomial selector activates one specific $p_i(x)$ generating the tap of the feedback polynomial. Representation with logic gates.

we based our estimations on previous works presented in [10], [12]. This way, the physical source of randomness that we assume consists of the thermal-noise oscillator presented by Che *et al.* in [8], but specified and modeled in our work as proposed in [13] and [14].

The remainder GEs mainly consist of the necessary extra circuitry for controlling the different states of the generator. The final amount of GEs is of, at most, about 453 logic gates. This perfectly matches the Gen2 requirements.

C. Power Consumption

The energy used for a (cryptographic) operation depends on the average power (P_{avg}) and the duration t of the computation. Hence, energy consumption is one of the most important parameters for battery-powered devices, which depends on the time that the battery is able to supply the requested power. However, for passively powered devices (such as EPC Gen2 tags) the average power transmitted from the reader to the tag is small but, in general terms, the reader can supply the power arbitrarily long. Thus, the energy does not play an important role in the EPC Gen2 tags as long as there is enough time available to do the computation [15].

TABLE VI
LOGIC GE COUNT FOR OUR PROPOSED PRNG

Element	Function	GE count
	LFSR	**73.3**
16 FF	16-bit Register	72
1 AND	PRNG output	1.3
	Polynomial Selector	**213.6**
15 AND + 15 XOR + 90 OR	LFSR Feedback	180
14 AND + 3 INV + 3 FF	Decoder	33.6
	Decoding Logic	**68**
6 FF	64 Cycle Clock	27
5 AND + 2 OR + 2 INV + 1 XOR	LFSR & Memory Clocking	13
4 FF + 4 AND + 1 OR + 1 INV	Feedback Selector Cycle	28
	Thermal-noise TRNG	**22**
	Additional control	**76**
	Total	**453**

Standard CMOS transistors is the current choice of most digital circuit designs built for low power consumption and robustness. Hence, it is appropriate to consider power consumption analysis based on an implementation using CMOS technology [10].

An important aspect of security implementations in the design stage is to ensure that the power dissipation of the IC does not exceed the available power budget for its execution. Feldhofer *et al.* have estimated the average power budget for cryptographic operations in 4 μW at five meters to the reader [15].

The total power consumption of a CMOS circuit is the sum of static and dynamic power consumption. The static power consumption mainly depends on the size of the circuit and is very small, thus, it can be ignored for our considerations [15]. While the use of direct methods to measure power dissipation may be possible, a simple method for estimating the dynamic power dissipation is based on formulating the power loss during the charging and discharging of capacitances [10]. Equation 1 models the average power dissipation of a system composed of a small number of logic gates.

$$P = p_{0\to1} C_L V_{DD}^2 f_{clk} \tag{1}$$

C_L is the load capacitance along the critical path and $p_{0\to1}$ represents the logic state transition from low to high (or vice versa) in a single clock cycle. The combination of $p_{0\to1}$ and C_L can also be stated as the average capacitance switched during each clock cycle. f_{clk} represents the clock frequency and V_{DD} is the system supply voltage. Design measures for lowering the power consumption result from minimizing the factors in this equation. It is difficult to apply this formula to ICs of large sizes due to the difficulty to state the number of logic state transitions for each clock cycle. However, it is adequate for estimating the power consumption in small hardware.

Based on measurements presented by Etrog *et al.* [16], the load capacitance (C_L) for each GE is approximately 3 fF. The voltage source and operating frequency have been previously stated in 1 V and 100 kHz for passive low-cost RFID. If we consider that about half GE are switched for each clock cycle, Equation 1 returns an estimation of 67.5 nW of average power consumption for our PRNG proposal. The estimation is consistent with similar designs present in the literature [17], and under the available budget of 4 μW of power consumption for cryptographic operations for UHF technologies.

V. RELATED WORK

Very few PRNGs designs for lightweight RFID technologies have been disclosed in the related literature. Manufacturers of existing commercial solutions are, indeed, reluctant to provide their designs [2]. The use of security through obscurity, as the case of the MIFARE RFID Classic chip has shown recently [3], is always ineffective in security engineering. Moreover, most of the designs that do appear in the literature, and that claim to be both secure and lightweight enough to fit the EPC

Gen2 restrictions, fail to provide convincing proofs of such claims. Some proper examples are [18] and [8]. The design in [18] is an optimized variant of the shrinking generator [19], a well studied cryptographic design that combines two clocked LFSRs. The output sequence of the first LFSR is used to discard some bits from the output sequence of the second LFSR [7]. The hardware implementation of this generator seems to require 1,435 logic gates. However, it is worth pointing out that some techniques presented in [20] can be used to attack the scheme. No evidences of how their proposal controls either the irregularities of the generator's output rate (an important drawback inherent to any shrinking generator scheme). If this problem is not properly handled, it can hint at the state of the main LFSR, and so breaking the security of the generator. The second example, presented in [8], is also a variant of the shrinking generator discussed above, but based on a physical source of randomness that handles the linearity of an underlying LFSR. In [21], [22], we presented an efficient attack for successfully retrieving the feedback polynomial of this vulnerable generator scheme with very few observations. Assuming a 16-bit version of the generator, we proved that the feedback polynomial can be predicted with a probability higher than 50% by simply capturing 160 bits; and 90% by capturing 464 bits. Therefore, the scheme does not meet any security standard.

TABLE VII
GE COMPARISON OF LIGHTWEIGHT PRNG PROPOSALS

	Trivium	LAMED	Grain	Our Proposal
GE Count	1,857	1,585	1,294	453

Some PRNGs based on stream ciphers schemes, such as Trivium [23], LAMED [2], or Grain [24] are also lightweight enough to satisfy the EPC Gen2 requirements. Up to now, no attacks against these three proposals exist in the literature. Table VII compares the hardware complexity (in GEs) of these proposals with ours, and shows that their complexity seems to be considerably much higher than ours. Other technical restrictions, such as power consumption, cannot be compared with ours since no evidences of such evaluations are provided in their work. However, and considering their hardware complexity, we can predict that it is also much higher than ours.

VI. Conclusion

A pseudorandom number generator (PRNG) design for EPC Gen2 RFID tags has been presented. The generator is based on a linear feedback shift register (LFSR) configured with a multiple-polynomial tap architecture that is fed, in turn, by a physical source of randomness. This leads to having different feedback primitive polynomials selected at random in order to handle the natural linearity of any LFSR. We have validated that the resulting generator satisfies the randomness requirements imposed by the EPC Gen2 standard. We have also evaluated the hardware complexity and the power consumption of our generator. The evaluations have confirmed that the proposal fulfills all the constraints imposed by the EPC Gen2 standard by using, moreover, a much simpler design than any other previous scheme reported in the literature.

References

[1] EPCglobal, "EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860-960 MHz," [Online] Available at `http://www.epcglobalinc.org/standards/`.

[2] P. Peris-Lopez, J. Hernandez-Castro, J. Estevez-Tapiador, and A. Ribagorda, "LAMED a PRNG for EPC Class-1 Generation-2 RFID specification," *Computer Standards & Interfaces*, pp. 88–97, Dec. 2007.

[3] F. Garcia, G. Koning, R. Muijrers, P. van Rossum, R. Verdult, R. Wichers, and B. Jacobs, "Dismantling MIFARE Classic," in *Computer Security - ESORICS 2008*, 2008, pp. 97–114.

[4] S. Hellebrand, J. Rajskia, S. Tarnick, S. Venkataraman, and B. Courtois, "Built-in test for circuits with scan based on reseeding of multiple-polynomial linear feedback shift registers," *Computers, IEEE Transactions on*, vol. 44, no. 2, pp. 223–233, Feb. 1995.

[5] P. Rosinger, B. Al-Hashimi, and N. Nicolici, "Dual multiple-polynomial LFSR for low-power mixed-mode BIST," *Computers and Digital Techniques, IEEE Proceedings on*, vol. 150, no. 4, pp. 209–217, Jul. 2003.

[6] J. Garcia-Alfaro, M. Barbeau, E. Kranakis. "Secure localization of nodes in wireless sensor networks with limited number of truth tellers," in *Communication Networks and Services Research Conference, (CNSR'09)*, 2009, pp. 86–93.

[7] A. Menezes, P. V. Oorschot, and S. Vanstone, *Handbook of applied cryptography*. CRC Press, Fifth Printing, 2001.

[8] W. Che, H.Deng, X. Tan, and J. Wang, *Networked RFID Systems and Lightweight Cryptography, Chapter 16*. Springer, Nov. 2008, ch. A Random Number Generator for Application in RFID Tags, pp. 279–287.

[9] M. Haahr, "True random number service," 2010, (Last access Feb. 2010). [Online]. Available: http://www.random.org

[10] D. C. Ranasinghe and P. H. Cole, *Networked RFID Systems and Lightweight Cryptography, Chapter 8*. Springer, Nov. 2008, ch. An Evaluation Framework, pp. 157–167.

[11] R. Baker, *CMOS: Circuit design, layout, and simulation*. Wiley-IEEE Press, 2007.

[12] C. Paar, A. Poschmann, and M. Robshaw, "New Designs in Lightweight Symmetric Encryption," in *RFID Security*, P. Kitsos and Y. Zhang, Eds. Springer US, 2009, pp. 349–371.

[13] C. Petrie and J. Connelly, "Modeling and simulation of oscillator-based random number generators," in *Circuits and Systems, IEEE International Symposium on*, vol. 4, May 1996, pp. 324–327.

[14] S. Zhou, W. Zhang, and N. Wu, "An ultra-low power cmos random number generator," *Solid-State Electronics*, 2008.

[15] M. Feldhofer and J. Wolkerstorfer, "Hardware Implementation of Symmetric Algorithms for RFID Security," in *RFID Security*, P. Kitsos and Y. Zhang, Eds. Springer US, 2009, pp. 373–415.

[16] J. Etrog, M. Robshaw, and O. Savry, "The possibilities and limitations of cryptography in constrained devices," INRIA and Orange Labs, Tech. Rep., 2009, deliverable for RFID-AP.

[17] A. Boni and A. Facen, "Ultra low-voltage analog circuits for UHF RFID devices in 180 nm CMOS technology," *Analog Integrated Circuits and Signal Processing*, vol. 63, no. 3, pp. 359–367, 2010.

[18] H. Lee and D. Hong, "The tag authentication scheme using self-shrinking generator on RFID system," *International Journal of Applied Science, Engineering and Technology*, vol. 3, pp. 33–38, 2007.

[19] D. Coppersmith, H. Krawczyk, and Y. Mansour, "The shrinking generator," in *Advances in Cryptology - Crypto'93*. Springer, 1994, pp. 22–39.

[20] W. Meier and O. Staffelbach, "The self-shrinking generator," in *Advances in Cryptology - EUROCRYPT'94*. Springer, 1995, pp. 205–214.

[21] J. Melia-Segui, J. Garcia-Alfaro, and J. Herrera-Joancomarti, "Analysis and Improvement of a Pseudorandom Number Generator for EPC Gen2 Tags," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, R. S. et al., Ed. Springer, 2010, vol. 6054, pp. 34–46.

[22] J. Melia-Segui, J. Garcia-Alfaro, and J. Herrera-Joancomartí, "A Practical Implementation Attack on Weak Pseudorandom Number Generator Designs for EPC Gen2 Tags," *Wireless Personal Communications*, 2010, DOI: 10.1007/s11277-010-0187-1.

[23] C. De Canniere and B. Preneel, “Trivium specifications,” Ecrypt, Tech. Rep., 2008. [Online]. Available: `http://www.ecrypt.eu.org/stream/triviump3.html`

[24] M. Hell, T. Johansson, and W. Meier, “Grain: a stream cipher for constrained environments,” *International Journal of Wireless and Mobile Computing*, vol. 2, no. 1, pp. 86–93, Nov. 2007.