Analysis and Improvement of a Pseudorandom Number Generator for EPC Gen2 Tags

J. Melia-Segui[1], J. Garcia-Alfaro[1,3], and J. Herrera-Joancomarti[2]

[1] Universitat Oberta de Catalunya,
Rambla Poble Nou 156, 08018 Barcelona - Spain,
melia@uoc.edu

[2] Universitat Autònoma de Barcelona,
Edifici Q, Campus de Bellaterra, 08193, Bellaterra - Spain,
jherrera@deic.uab.es

[3] Institut Telecom, Telecom Bretagne
02, rue de la Chatagneraie, Cesson-Sevigne 35576 - France
joaquin.garcia-alfaro@acm.org

Abstract. The EPC Gen2 is an international standard that proposes the use of Radio Frequency Identification (RFID) in the supply chain. It is designed to balance cost and functionality. The development of Gen2 tags faces, in fact, several challenging constraints such as cost, compatibility regulations, power consumption, and performance requirements. As a consequence, security on board of Gen2 tags is often minimal. It is, indeed, mainly based on the use of on board pseudorandomness. This pseudorandomness is used to blind the communication between readers and tags; and to acknowledge the proper execution of password-protected operations. Gen2 manufacturers are often reluctant to show the design of their pseudorandom generators. Security through obscurity has always been ineffective. Some open designs have also been proposed. Most of them fail, however, to prove their correctness. We analyze a recent proposal presented in the literature and demonstrate that it is, in fact, insecure. We propose an alternative mechanism that fits the Gen2 constraints and satisfies the security requirements.

1 Introduction

The EPC Gen2 is an international standard that proposes the use of Radio Frequency Identification (RFID) in the supply chain. It is designed to balance cost and functionality. The development of Gen2 tags faces, in fact, several challenging constraints such as cost, compatibility regulations, power consumption, and performance requirements. As a consequence, the computational capabilities of Gen2 tags are very simple. In this sense, the Gen2 specification only considers two basic on board security features: pseudorandom number generators (PRNGs) and password-protected operations. The pseudorandomness offered by on board PRNGs is, indeed, used to protect the password-protected operations. PRNGs are also used as an anti-collision mechanism for inventorying processes [4]; and to acknowledge other Gen2 specific operations (e.g., memory writing, decommission of tags, and self-destruction). PRNGs are, therefore, the crucial components that guarantee Gen2 security.

Commercial developments of the Gen2 standard are often reluctant to present the design of their PRNGs. Manufacturers simply refer to testbeds that show the accomplishment of some expected requirements, most of them for compatibility purposes. They fail to offer convincing information about the PRNGs designs [15]. This is mostly security through obscurity, which is always ineffective in security engineering. Vulnerable designs appeared in recent commercial RFID technologies, such as the vulnerable PRNGs used by the cryptosystem of the MIFARE Classic chip [5], confirm this principle. Cryptographic suitable PRNGs designs must, moreover, satisfy unpredictability characteristics. For example, an external adversary who eavesdrops the communication cannot compute the PRNG internal state, even if many outputs of the generator have been observed. The adversary cannot either compute the next sequence, even if many other previous sequences have been observed. If the adversary can observe, or even manipulate, the input samples that are fed by a PRNG, but its internal state is not known, the adversary must not be able to compute the next sequence or the next internal state of the PRNG. Finally, if the adversary has somehow learned the internal state of the PRNG, but the input samples that are fed in cannot be observed, then the adversary should not figure out the internal state of the PRNG after the re-keying operation. Most of these characteristics are, in fact, required by the EPC Gen2 specification [4].

PRNGs designs for highly resource-constrained devices (e.g, Gen2 RFID tags) exist in the literature (e.g., [11, 10, 1, 14, 2]). Some of them fail, however, to proof their correctness. We analyze in this paper the approach presented in [2], in which Che *et al.* propose the use of *linear feedback shift registers* (LFSRs) fed by an oscillator-based physical device that transforms thermal noise into *true random* sequences of bits. The authors claim that this approach leads to the construction of cost-effective PRNGs for RFID devices. For example, a Gen2 compatible PRNG can be implemented by using a 16-bit LFSR that is modified on every interrogation by XORing some of the LFSR cells with the random bits of the oscillator-based device. We demonstrate, however, that their approach leads to insecure implementations. We proof that the scheme does not succeed in handling the linearity of LFSRs. We show how an eavesdropper may obtain the feedback polynomial of the LFSR by using very few observations. We propose, moreover, an alternative solution that highly improves the security of the analyzed scheme. Our improvement fits, moreover, the resource constraints of Gen2 devices.

Paper Organization — Section 2 describes the suitability of using LFSRs for the generation of pseudorandom sequences and analyzes the Che *et al.* scheme. Section 3 describes an attack to the scheme. Section 4 introduces an alternative solution. Section 5 surveys some related works.

2 LFSR-based Pseudorandom Number Generators

A linear feedback shift register (LFSR) is a digital circuit that contains a shift register and a feedback function. The shift register is composed of a sequence of binary cells that share the same clock signal. Each time a bit is needed, the content of the register is shifted one cell, obtaining the most significant bit of the register in the previous state. The feedback function computes a new bit using some bits of the register, obtaining the less significant bit to be filled in the new state of the register. The feedback function of an LFSR is basically an exclusive or logical operation (XOR, denoted as $\oplus$ hereinafter) of some cells content, named *taps*. The period (quantity of different possible states) of an LFSR with n cells is up to $(2^n - 1)$ when taps configuration follows a primitive-polynomial function, with optimum statistical properties, such as:

$$C(x) = 1 + c_1 x^1 + c_2 x^2 + \cdots + c_n x^n \tag{1}$$

The LFSR can then be determined by this polynomial function. In turn, the sequences of the LFSR can be determined by the polynomial function of the LFSR and the initial state of the register cells (often referred as *seed*).

LFSRs are the most common type of shift registers used in cryptography. They lead to efficient and simple hardware implementations. They have, however, important drawbacks that must be handled. First, the sequences of an LFSR are predictable [9, 3]. For example, let $s_{k+1}, s_{k+2}, \cdots, s_{k+2n}$ be a sequence of $2n$ consecutive bits generated from an LFSR. Let $c_n, c_{n-1}, \cdots, c_1$ be the feedback function of the LFSR. Then, the feedback function can be easily computed by solving the following equation system:

$$\begin{bmatrix} s_{k+1} & s_{k+2} & \cdots & s_{k+n} \\ s_{k+2} & s_{k+3} & \cdots & s_{k+n+1} \\ \vdots & \vdots & \ddots & \vdots \\ s_{k+n} & s_{k+n+1} & \cdots & s_{k+2n-1} \end{bmatrix} \begin{bmatrix} c_n \\ c_{n-1} \\ \vdots \\ c_1 \end{bmatrix} = \begin{bmatrix} s_{k+n+1} \\ s_{k+n+2} \\ \vdots \\ s_{k+2n} \end{bmatrix} \tag{2}$$

By solving Equation (2) we obtain the feedback polynomial coefficients. Therefore, a n-bit (cells) LFSR with period $2^n - 1$ can be determined with only $2n$ values. This linearity must be handled before using LFSRs to build *pseudorandom number generators* (PRNGs). There are several solutions in the literature to decrease the degree of linearity of LFSR-based PRNGs. The use of non-linear filtering and the combination of multiple LFSRs are appropriate examples. Another way of decreasing the linearity degree of LSFR-based PRNGs is the addition of true random bits to the feedback function. This is in fact the strategy proposed by Che *et al.* in [2] for the construction of a cost-effective PRNG for RFID devices. We analyze their proposal in the sequel.

2.1 Che *et al.* Scheme Brief Description

The combination of true random numbers (*trn*) and PRNG techniques are used when *trn* generation throughput is not enough to cover the stream generation requirement. The *trn* is therefore used for replacing some parts of the PRNG stream or as a seed for PRNG initialization. Although *trn* addition can also be applied to LFSRs in PRNG, there are not many references regarding this technique in the literature. This is because *trn* addition to PRNG communication model cannot be applied to a traditional communication scheme where *sender* and *receiver* share k as a key for the PRNG one-time pad transmission/reception, because of the uncertainty of the *trn*. On the other hand, *trn* addition to PRNG is of a great interest for RFID communications where good PRNG are needed for secured communications. Specially in the EPC Gen2 technology, where the usage scenario does not allow the key sharing [4].

Che *et al.* present in [2] a new PRNG for application in RFID tags, improving the poor randomness from the basic PRNGs. This mechanism relies on an oscillator-based

Truly RNG (TRNG), and exploits the thermal noise of two resistors to modulate the edge of a sampling clock. Authors state the final system prevents potential attackers to perform any effective prediction about the generated sequence (even if the design is known) thanks to the white noise based cryptographic key generation.

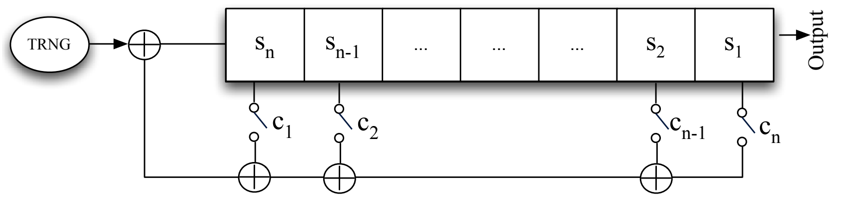

Fig. 1. PRNG scheme based on the Che *et al.* specifications.

After describing its TRNG oscillator-based core, the authors focus on design considerations specially regarding *power consumption* and *output data rates* trade-offs. Knowing the fact that the higher the frequency oscillation of the system, the higher the current (thus also power) consumption, the authors look for system level optimization in order to reduce the power consumption due to the low-power restrictions of RFID.

The optimization proposed by Che *et al.* relies on the combination of the TRNG and a *linear feedback shift register* (LFSR) (cf. Figure 1). Adding an LFSR to the TRNG lets the system reduce the clock frequency proportionally to the number of cells of the LFSR. Specifically, exploiting the initial state of a 16-bit LFSR combined with the addition of the generated truly random number (*trn*) for each cycle ring, allows the system to decrease the clock frequency with a $\frac{1}{16}$ factor.

According to the authors, the addition of only a truly random bit in the cycle ring as a random number seed, the LFSR output sequence will be unpredictable and irreproducible, just like a TRNG. We show in the sequel that this claim is false.

2.2 Predictability of the Scheme

We have detailed above that the main vulnerability of a PRNG based on a linear feedback register comes from its easy predictability due to its linearity properties. We will show that the randomness introduced in the Che *et al.* scheme is not enough to mask the linearity of the scheme.

Following the Che *et al.* scheme (cf. Figure 1) the pseudorandom sequence is produced by an LFSR XORed in its first cell with a *truly* random bit (generated in the oscillator) for each register cycle in order to be unpredictable and irreproducible [2]. The pseudorandom output sequence for an n cell LFSR can be represented as:

$$s_{k+1} \oplus trn_1, s_{k+2} \oplus trn_1, s_{k+3} \oplus trn_1, \dots s_{k+n} \oplus trn_1,$$

$$s_{k+n+1} \oplus trn_2, \dots s_{k+2n} \oplus trn_2, s_{k+2n+1} \oplus \dots$$

Since the LFSR seed is modified with the trn_i bit, the LFSR output will also be modified regarding the trn values. If we assume that the trn_i bits are generated by a true random generator, then the probability that $trn_i = 0$ or $trn_i = 1$ is equal to $p = \frac{1}{2}$. Then, since the trn_i value is only XORed for each cycle, when two consecutive 0's are generated by the true random generator, $trn_i = trn_{i+1} = 0$, then the $2n$ bits output stream of the system will be exactly the same of the one produced by the LFSR. This situation can represent a threat for the unpredictability of the system, since these $2n$ values can be used to obtain the feedback polynomial of the LFSR.

3 Proposed Attack

Based on the vulnerability sketched in the previous section, we present a detailed attack on the Che *et al.* scheme. Our scenario is composed by a Che *et al.* system that produces pseudorandom bits. Only a part of the pseudorandom output sequence, denoted by s_a, is known to the attacker. The attack will succeed if the attacker can provide the LFSR feedback polynomial. From now on, we denote by $|s_a|$ the length of s_a.

To generalize the attack, we also assume that the attacker cannot determine the first bit of the sequence, that means he has no information if a given s_a sequence, with $|s_a| = 2n$, has been affected by exactly two trn values (that means the attacker finds two exact LFSR periods) or the sequence has been modified by three trn values.

With these constrains, given a sequence, s_a with $|s_a| = 2n$, the probability that s_a has been affected by exactly two trn is $\frac{1}{n}$. Furthermore, the probability that the two trn used in that sequence are exactly zeros is $\frac{1}{4}$. Then, given $|s_a| = 2n$ from a Che *et al.* output sequence if we analyze the system as described in Section 2 we will obtain the correct feedback polynomial with probability $\frac{1}{4n}$. However, in this situation, the attack itself cannot verify the correctness of the resulting polynomial.

Now, assume that $|s_a| = 3n - 1$. If the sequence is divided into n subsequences of length $2n$, we can ensure that one of these subsequences has been affected by exactly two trn. The remainder $n-1$ subsequences, have been affected by three trn. However, notice that if the three trn are zeros, the n vectors of length $2n$ will give the same feedback polynomial. The probability of such event is $\frac{1}{8}$. Then, Equation 3 provides the probability of success of an attack that analyzes a sequence with $|s_a| = 3n - 1$:

$$P_{success}(3n-1) = \frac{1}{4}\left(\frac{1}{n}\right) + \frac{1}{8}\left(\frac{n-1}{n}\right) = \frac{n+1}{8n} \tag{3}$$

Furthermore, in this case where $|s_a| = 3n - 1$ the attack is self-verified since all n vectors will produce the same feedback polynomial, and then, the attacker will be sure to have obtained the correct polynomial.

Notice that $3n - 1$ is the smaller sequence that produces a self verified attack in the sense that n identical feedback vectors are found, providing three consecutive zeros in the true random sequence. Obviously, the probability of success increases with $|s_a|$ since increasing the $|s_a|$ implies that more trn bits affect the sequence and then the probability of finding three consecutive zeros also increases.

Figure 2 shows the probability of success of an attack with s_a length for a particular system with an LFSR of length $n = 16$, like in the Che *et al.* scheme [2] and the EPC

Gen2 specifications. Notice that only 160 bits ($10n$) are enough to perform a successful attack with probability higher than 50%, and 464 bits ($29n$) implies more than a 90% of success probability.

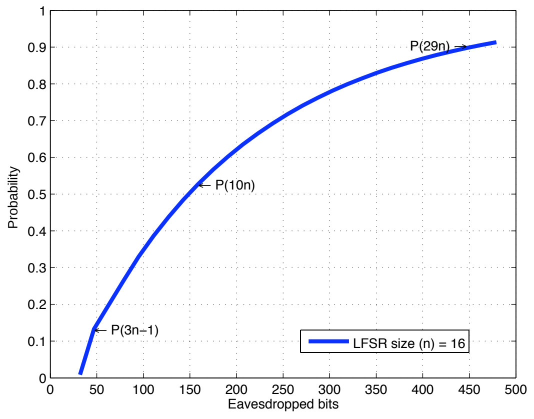

Fig. 2. Reliability on the Che *et al.* attack regarding $|s_a|$.

3.1 Attack Implementation

The proposed attack defined above has been implemented to support the theoretical analisys with practical results.

The Che *et al.* scheme has been implemented by strictly following the specifications stated in [2]. The code has several configurable parameters, such as the size of the LFSR, the feedback polynomial, and the seed values. The sequences of true random bits are obtained from [6]. Algorithm 1 provides the pseudocode of the attack.

Ten different test sequences of 341 MB, T_i, have been generated with different *seed* and true random bit sequences. Several experiments have been performed over each generated sequence T_i. Two different analysis have been done. The first one validates that the probability of finding the feedback polynomial matches the one described in

Algorithm 1 Attack to the Che *et al.* Model.

```
1:  $count \leftarrow 0$; // Initialize counter
2:  // Initialize index i at a random position
3:  // data_set stores $2^n - 1$ bits of data
4:  While $count < size(LFSR)$ do
5:      take $vector[i \,..\, 2n + i]$ from $data\_set$;
6:      compute $polynom$ from $vector$; // cf. Equation 2
7:      If $(polynom_{prev.} = polynom)$ then
8:          $count \leftarrow count + 1$;
9:      Else
10:         $count \leftarrow 0$;
11:     $i \leftarrow i + 1$;
12:     $polynom_{prev.} \leftarrow polynom$;
```

Equation 3. In this case, the algorithm takes $|s_a| = 3n - 1$ bits from T_i starting at a random position and tries to attack the system by finding n equal feedback polynomials. The operation is repeated one thousand times for each test sequence T_i. Attack success rates are reported in Table 1. Notice that they are close to the theoretic value $\frac{(n+1)}{8n}$ with $n = 16 \approx 0,1328$.

The second analysis provides the number of bits that has been needed to achieve a successful attack. Ten different attacks have been performed for every T_i data sequence taking the first bit of s_a at random. Results presented in Table 2 show the number of bits for a successful attack in the worst case, that is the attack that needs a major number of bits. Notice that, although taking the worst case, the number of bits is significantly lower than the whole period $2^{16} - 1$.

Sequence	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
% of attack success	0.1320	0.1370	0.1310	0.1260	0.1390	0.1370	0.1290	0.1370	0.1380	0.1280

Table 1. Attack success rate for $|s_a| = 3n - 1$.

Sequence	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
$\lvert s_a \rvert$	238	254	254	190	510	158	254	286	238	222

Table 2. Value of $|s_a|$ for a successful attack in the worst case after 10 tests.

4 Proposed PRNG Scheme

We present a new PRNG scheme based also on the use of a LFSR, and perturbed by true random data. Our proposal successfully handles the vulnerabilities found in the Che *et al.* scheme [2]. We show, moreover, that our proposal is compatible with the requirements defined by EPCglobal for designing Gen2 compliant PRNGs [4].

4.1 System Description

Similarly to the Che *et al.* scheme, our proposal relies on a *linear feedback shift registers* LFSR core perturbed by a *true random number* (*trn*) source. We keep the LFSR core for different reasons. On the one hand, LFSR schemes are very fast and efficient in hardware implementations as well as simple in terms of computational requirements. This makes the use of LFSRs an ideal system for both energy and computational constrained environments. On the other hand, an LFSR follows the same hardware scheme than *cyclic redundancy check* (CRC) functions. These functions are included in the EPC Gen2 standard. Therefore, current EPC Gen2 tags including CRC are able of executing LFSR-based functions in the same hardware.

Different proposals exist to derive true random sequences of bits from the hardware of an RFID tag. Some examples of on-tag trn acquisition are, for instance, taking advantage of thermal noise, high frequency sampling or fingerprint data in circuits. Some commercial tags include, moreover, some extra functionalities (e.g., *received signal strength indicator*, RSSI [13]) that can be useful for *trn* addition techniques.

Similarities of our scheme with the one of Che *et al.* end here. In our proposal, randomness is used in a different way in order to truly mask the linearity of the LFSR. We have seen in the Che *et al.* scheme that using true random data to modify the output of the LFSR is not enough to break the predictability of the LFSR. We take a different approach and we use the *trn* bits to modify the characteristic polynomial of the LFSR rather than the LFSR output. A first idea is to replace the static feedback polynomial

$$C(x) = 1 + c_1x^1 + c_2x^2 + \cdots + c_nx^n$$

with a dynamic one, that depends on the true random data

$$C(x) = 1 + (trn_j)x^1 + (trn_{j+1})x^2 + \cdots + (trn_{j+n})x^{n-1} + x^n$$

where only the most significant cell is always switched on to set the function degree to n. However, such an approach does not produce a good pseudorandomness output sequence since not all feedback polynomials randomly generated are primitive. Feedback polynomials of an LFSR must be primitive to guarantee good pseudorandom properties. Using primitive polynomials as feedback polynomials must, therefore, be enforced.

Taking different primitive polynomials as the feedback polynomial of an LFSR has already been used in non-security related scenarios. In [8, 17], for instance, the authors call this technique *Multiple-Polynomial* (MP) LFSRs. They apply this technique for *Built-In Self Tests* (BIST) operations. These operations are intended for testing chip designs, generating test vectors and evaluating test responses. The *multiple-polynomial* characteristic means several polynomial configurations are applied to the LFSR, depending on an input parameter. These schemes must guarantee complete fault coverage tests while minimizing test application time, test overhead and data storage [17].

Following these ideas, we build up our PRNG design using an LFSR that is enhanced by a multiple feedback polynomial. Instead of a fixed feedback polynomial, the LFSR uses 2^m different feedback primitive polynomials. A *decoding logic unit* provides, at every LFSR cycle, one of the 2^m primitive polynomials as a feedback polynomial. The selection of each primitive polynomial for every cycle is performed by the true random data source. We present in the sequel the implementation details of our proposal. We discuss the exact parametrization of the system and provide some practical results.

4.2 Implementation Details

We fix the length of the LFSR to $n = 16$. This value offers EPC Gen2 tag compatibility and allows a better comparison with the proposal of Che *et al.* The total number of different feedback polynomials is set to eight (i.e., indexed by three *trn* bits). This value gives an appropriate trade off between computational and system complexity. Although an increase of the number of feedback polynomials leads to a higher number of different primitive polynomials, it also increases the amount and complexity of logial gates onboard of the tag. It is assumed that the price of a given circuit increases by one cent for each extra one thousand gates [12].

Primitive polynomials
$x^{16} + x^{15} + x^{10} + x^9 + x^8 + x^6 + 1$
$x^{16} + x^{15} + x^{14} + x^{12} + x^{10} + x^9 + 1$
$x^{16} + x^{15} + x^{14} + x^{10} + x^9 + x^8 + 1$
$x^{16} + x^{15} + x^9 + x^6 + 1$
$x^{16} + x^{15} + x^9 + x^4 + 1$
$x^{16} + x^{15} + x^{12} + x^9 + x^6 + x^4 + 1$
$x^{16} + x^{15} + x^{14} + x^{12} + x^9 + x^8 + 1$
$x^{16} + x^{15} + x^{14} + x^{12} + x^9 + x^4 + 1$

Table 3. Feedback polynomials used in our scheme.

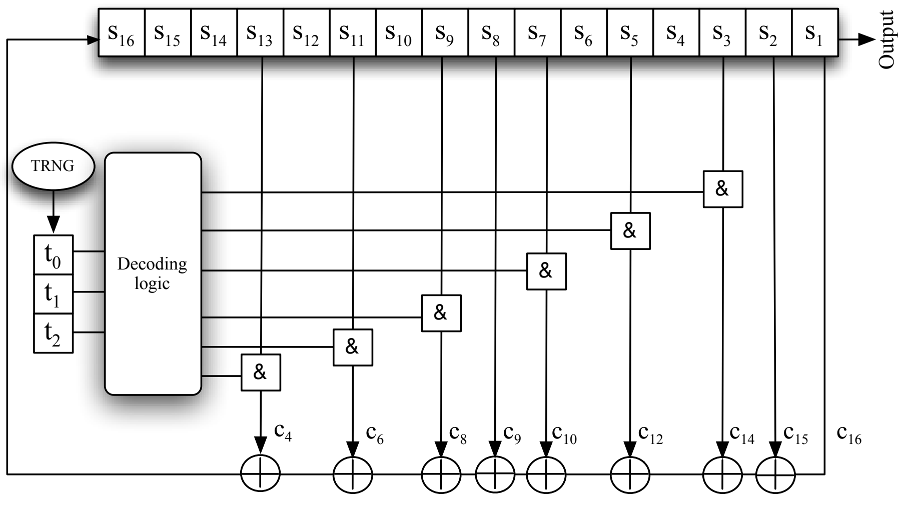

Fig. 3. Gen2 compliant PRNG proposal.

The selected polynomials, included in Table 3, are primitive polynomials of degree 16 with the highest number of common elements. From 2,048 possible primitive polynomials, the selected ones have ten common elements and six different ones. With this special selection only six bits are needed to encode all of them. To avoid two consecutive selections of the same feedback polynomial, what would turn into a prediction vulnerability, a simple rotation is applied to the *decoding logic unit* for the polynomial selection. Regarding this polynomial selection, Figure 3 shows our proposed system with polynomial tap configurations.

4.3 Suitability to the EPC Gen2 Standard

The proposed PRNG system has been implemented in a software simulation in order to check its suitability as a PRNG for EPC Class1 Gen2 standard [4]. The pseudorandom datasets have been obtained from our PRNG using the same initial parameters (seed and *trn* source) than the ones used in Section 3.1. We generated above 3.3 Gb of data, divided in ten test sequences T_i. This amount of data represents 1.5 hours of constant 16-bit numbers transmission, assuming a bit rate of 640 kbps (as it is specified by the EPC Gen2 standard).

Statistical behaviour — The EPC specification defines three statistical properties that PRNGs on board of Gen2 tags must satisfy:

1. **Probability of a single sequence —** The probability that any random sequence drawn from the PRNG has value j, for any j, shall be bounded by:
$$\frac{0.8}{2^{16}} < P(j) < \frac{1.25}{2^{16}} \tag{4}$$
2. **Probability of simultaneously identical sequences —** For a tag population of up to ten thousand tags, the probability that any of two or more tags simultaneously generate the same sequence of bits shall be less than 0.1%, regardless of when the tags are energized.
3. **Probability of predicting a sequence —** A given sequence drawn from the PRNG 10 ms after the end of the transmission shall not be predictable with a probability greater than 0.025% if the outcomes of prior draws from PRNG, performed under identical conditions, are known.

Regarding the first property, tests over the generated data checked the occurrence of each 16-bit values. The obtained values for the ten test sequences, included in Table 4, show that after almost 200 million of sequences were analyzed, the probability of occurrence of a 16-bit value lies between $\frac{0{,}90}{2^{16}}$ and $\frac{1{,}09}{2^{16}}$. Then, our proposed PRNG fulfills the first specification of the EPC Gen2 standard.

The second property for building Gen2 compliant PRNGs enforces that two simultaneous identical sequences must not appear with more that 0.1% for a population up to ten thousand tags. To test this property, ten thousand PRNGs have been initialized with random data in order to simulate a real population of 10,000 tags. The correlation of the ten thousand obtained sequences has been performed. Due to the true random data that uses the proposed system, none of the different systems generate the same sequence.

The third property is related to the probability of prediction, stating that a 16-bit pseudorandom number shall not be predictable with a probability greater than 0.025%, if the outcomes of prior draws from PRNG performed under identical conditions are known. Since our scheme uses a trn input to generate the output sequence, predictability becomes very difficult. To prove further, a serial correlation test has been performed. This test computes the degree of dependence of a *n* bit output from the previous one. Results, shown in Table 5, are very close to zero which determines good pseudorandomness.

Sequence	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
Lowest $\frac{p}{2^{16}}$	0.9246	0.9199	0.9082	0.9169	0.9151	0.9217	0.9195	0.9191	0.9184	0.9246
Highest $\frac{p}{2^{16}}$	1.0792	1.0821	1.0850	1.0781	1.0839	1.0832	1.0861	1.0799	1.0869	1.0811

Table 4. Successful fulfillment of our proposal to the first requirement of the EPC Gen2 standard.

Sequence	T_1	T_2	T_3	T_4	T_5
Correlation	-0.000046	-0.000005	-0.000038	-0.000023	0.000036
Sequence	T_6	T_7	T_8	T_9	T_{10}
Correlation	-0.000036	0.000000	0.000025	-0.000055	-0.000089

Table 5. Successful fulfillment of our proposal to the third requirement of the EPC Gen2 standard.

Hardware constrains — Once we have checked the PRNG proposal suitability to Gen2 in terms of statistical behavior, we analyze now some hardware related issues. Specifically, pseudorandom generators for the EPC Gen2 standard are expected to be implemented with a small amount of equivalent logic gates, defined in the literature between 2,000 and 5,000 [16]. The available time for a label operation in real-time is also of major importance. This value will condition the PRNG complexity, regarding the hardware scenario constraints. According to [4], the maximum tag to reader (uplink) data transmission rate is 640 kbps. Some authors place the PRNG execution time between 5 and 10 ms taking as a reference the performance criteria of an RFID system that demands a minimum label reading speed of at least 200 labels per second [16], or 2.2 ms taking as a reference the system clock frequency $f_S = 100$ KHz (that implies a clock cycle of 0.01 ms) by reading 450 tags in one second [14].

Element	**Function**	**Gate count**
LFSR16	Register for PRNG output	192
LFSR3	Register for trn storage	36
6 AND	For feedback polynomial selection	15
8 XOR	XOR operations	20
Decoding logic	MUX selection and rotation logic	347
Seed storage	For initialization purposes	24
Control (20%)		127
Total		761

Table 6. Logical Gate Equivalence for our Proposed PRNG.

Regarding existing estimations presented in the literature (e.g., [16, 14, 7]), we approximate the hardware complexity of our approach in 634 logic gates. Adding a 20%

of logic gates for control purposes as recommended in [14], the final amount is of 761 logic gates (cf. Table 6). This value perfectly matches the Gen2 requirements, and it has a lower hardware complexity than other low-overhead PRNG proposals for RFID as LAMED or Grain [14, 7]. For the time consumption requirement, taking the most restricting criteria that forces the generation of 16-bit sequences in 220 clock cycles (2.2 ms) [14], our proposal remains suitable enough for the generation of sixteen LFSR rotations and feedback polynomial selection.

5 Related Works

Some proposals in the literature propose suitable PRNG designs for Gen2 tags. We specially focus on designs motivated by security purposes. In this sense, Peris-Lopez *et al.* present in [14] a deterministic algorithm that relies on the use of 32-bit keys and pre-established initial states. Similarly, Klimov et al. present in [11] invertible bit transformations of 32 or 64 bits, suitable for PRNG applications. Other authors propose the use of on board physical properties to obtain random data generation. Holcomb *et al.* show in [10] a method to derive random data using the initial state of tag memory. Balachandran *et al.* propose in [1] the extraction of randomness by sampling radio signals. Che *et al.* describe in [2] an hybrid approach that combines the use of Linear Feedback Shift Registers (LFSR) and physical properties to build random sequences. We demonstrated in Section 2 that their approach is not secure, and presented in Section 3 an enhanced version based on a multiple-polynomial LFSR scheme [8, 17]. It is worth mentioning that Strüker *et al.* also cite in [18] functional weaknesses of the Che *et al.* scheme. Although, no results nor proofs are given in their paper.

6 Conclusions

We analyzed a *pseudorandom number generator* (PRNG) model for Radio Frequency Identification (RFID) devices, presented by Che *et al.* in [2]. The scheme uses a 16-bit *linear feedback shift register* (LFSR) for the generation of pseudorandom sequences. The LFSR is modified each cycle by XORing the first cell of the LFSR and a *true random* bit. We demonstrated that the proposal is not appropriate for security purposes, since it does not correctly handle the inherent linearity of the LFSR. We then showed empirically the possibility of successfully retrieving the feedback polynomial of the LFSR by using very few observations.

A new scheme has been then proposed. Our model is based on the use of a *multiple-polynomial* LFSR. We analyzed a 16-bit PRNG based on a software simulation of our model. We performed statistical analysis of random sequences generated by our simulation. Results confirm the validity of our technique. A hardware complexity estimation has also been presented. Our estimation successfully meets the requirements of the EPC Gen2 standard.

Acknowledgments — The 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 CSD2007-00004 ARES, an IN3-UOC doctoral fellowship, and the IRF project LOCHNESS.

References

1. G. Balachandran and R. Barnett. A 440nA true random number generator for passive RFID tags. *Circuits and Systems I: Regular Papers, IEEE Transactions on , vol.55, no.11, pp.3723-3732*, December 2008.
2. W. Che, H.Deng, X. Tan and J. Wang. In: *Networked RFID Systems and Lightweight Cryptography*, Chapter 16, *A Random Number Generator for Application in RFID Tags*, pp. 279–287. Springer, 2008.
3. C. L. Chen. Linear Dependencies in Linear Feedback Shift Registers. *Computers, IEEE Transactions on , vol.C-35, no.12, pp.1086-1088*, December 1986.
4. EPCglobal. EPC radio-frequency identity protocols class-1 generation-2 UHF RFID protocol for communications at 860-960 MHz. Tech. report, [On-line] `http://www.epcglobalinc.org/standards/`, 2007.
5. F. Garcia, G. Koning, R. Muijrers, P. van Rossum, R. Verdult, R. Wichers and B. Jacobs. In: *Computer Security - ESORICS 2008*, chapter *Dismantling MIFARE Classic*, pp. 97–114. Springer, 2008.
6. M. Haahr. True random number service. [On-line] `http://www.random.org`.
7. M. Hell, T. Johansson and W. Meier. Grain: a stream cipher for constrained environments. *International Journal of Wireless and Mobile Computing*, Volume 2, Issue 1, Pages 86-93, November 2007.
8. S. Hellebrand, J. Rajski, 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*, February 1995.
9. T. Herlestam. On Functions of Linear Shift Register Sequences. *Advances in Cryptology EUROCRYPT 85, LNCS , vol. 219/1986, DOI: 10.1007/3-540-39805-8, pp.119-129*, January 1995.
10. D. Holcomb, W. Burleson and K. Fu. Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. *Proceedings of the Conference on RFID Security*, July 2007.
11. A. Klimov and A. Shamir. In: *Cryptographic Hardware and Embedded Systems - CHES 2002*, chapter *A New Class of Invertible Mappings*, pp. 470–483. Springer, 2003.
12. M. Lehtonen, T. Staake, F. Michahelles and E. Fleisch. In: *Networked RFID Systems and Lightweight Cryptography*, chapter 9 *From Identification to Authentication - A Review of RFID Product Authentication Techniques*, pp. 169–187. Springer, November 2007.
13. Motorola. XR Series RFID Readers. Product Guide, [On-line] `https://docs.symbol.com/manuals/SIGN_71773.pdf`, 2008.
14. 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*, 2008.
15. P. Peris-Lopez. Lightweight Cryptography in Radio Frequency Identification (RFID) Systems. PhD Thesis, 2008.
16. D. Ranasinghe and P. Cole In: *Networked RFID Systems and Lightweight Cryptography*, chapter 8 *An Evaluation Framework*, pp. 157–167. Springer, November 2007.
17. P. Rosinger, B.M. Al-Hashimi and N. Nicolici. Dual multiple-polynomial LFSR for low-power mixed-mode BIST. *Computers and Digital Techniques, IEE Proceedings - , vol.150, no.4, pp. 209-217*, July 2003.
18. J. Strüker, C. Wonnemann, M. Kähmer and D. Gille Managing the Deactivation Process of EPC Class-1 Generation-2 Tags in Retail Industry. University of Freiburg, Germany, [Available On-line] `http://www.telematik.uni-freiburg.de`, 2007.