Sensors **2014**, *14*, 6500-6515; doi:10.3390/s140406500

OPEN ACCESS
sensors
ISSN 1424-8220
www.mdpi.com/journal/sensors

Article

EPCGen2 Pseudorandom Number Generators: Analysis of J3Gen

Alberto Peinado [1], Jorge Munilla [1,*] and Amparo Fúster-Sabater [2]

[1] Escuela Técnica Superior de Ingeniería de Telecomunicación, Universidad de Málaga, Málaga, Spain; E-Mail: apeinado@ic.uma.es

[2] Instituto de Tecnologías Físicas y de la Información, Consejo Superior de Investigaciones Científicas, Madrid, Spain; E-Mail: amparo@iec.csic.es

* Author to whom correspondence should be addressed; E-Mail: munilla@ic.uma.es; Tel.: +34-952-134-166.

Received: 18 December 2013; in revised form: 1 April 2014 / Accepted: 2 April 2014 / Published: 9 April 2014

Abstract: This paper analyzes the cryptographic security of J3Gen, a promising pseudo random number generator for low-cost passive Radio Frequency Identification (RFID) tags. Although J3Gen has been shown to fulfill the randomness criteria set by the EPCglobal Gen2 standard and is intended for security applications, we describe here two cryptanalytic attacks that question its security claims: (i) a probabilistic attack based on solving linear equation systems; and (ii) a deterministic attack based on the decimation of the output sequence. Numerical results, supported by simulations, show that for the specific recommended values of the configurable parameters, a low number of intercepted output bits are enough to break J3Gen. We then make some recommendations that address these issues.

Keywords: pseudo random number generators; security; cryptanalytic attack; Radio Frequency Identification; EPCglobal Gen2

---

1. Introduction

The term "Internet of Things" (IoT) was coined in 1999 by Kevin Ashton, a co-founder of the Auto-ID Center [1], a center that promoted the research and development of tracking products for supply-chains by using low-cost Radio Frequency Identification (RFID) tags. The advantages of RFID

over barcode technology are that it is wireless, does not require direct line-of-sight and that tags can be interrogated at greater distances, faster and concurrently [2]. This makes the IoT a wireless network of objects and sensors that collect and process information autonomously. RFID tags and sensors enable computers to observe, identify and understand for situational awareness without the limitations of human-entered data.

Nowadays, RFID is already a mature technology, and it is widely deployed for supply-chain, retail operations, inventory management and automatic identification in general. A typical RFID architecture involves three main components: (i) tags or transponders, which are electronic data storage devices that are attached to the objects to be identified; (ii) readers or interrogators, which manage the tag population, read data from and write data to tags; and (iii) a back-end server, which is a trusted entity that exchanges tag information with the readers and processes these data according to the specific intended application. Most tags are passive, which means that they do not have any kind of battery and receive the energy that they need to work from the reader. Thus, tags are inactive till they pass through the electromagnetic field generated by a reader, which is tuned to the same frequency.

The initial designs of RFID protocols focused on the performance with little attention being paid to resilience and security. However, as early as 2002, the first papers pointing out some possible security and privacy issues were already published, and in 2003, the CASPIAN (Consumer Against Supermarket Privacy Invasion and Numbering) group raised concerns regarding the possible misuse of RFID technology and called for boycotts against companies that decided to incorporate them. The European Commission in 2008 launched a public consultation on the issues of the use of RFID technology, particularly in terms of privacy, data protection and information security [3]. RFID can be indeed used to perform different forms of privacy invasion, such as unauthorized reading or tracking of people, and can be subject to impersonation. To overcome these issues, apart from the legal pressure for the protection of personal information (e.g., [4] in Europe and [5] in the U.S.), the technical means to control the access to tags is the implementation of cryptographic mechanisms that take into account their special characteristics: power-constrained devices, the vulnerability of the radio channel, reply upon-request, *etc*.

This increasing concern about security is evidenced with the inclusion of some optional cryptographic features in the recently ratified (November 2013) second version of the EPCglobal Gen2 Specification [6]. EPCglobal Gen2, hereafter EPCG2, is the standard (ISO [7]) for low-cost tags that work in the Ultra High Frequency (UHF) band of 860–960 MHz. This defines a platform for RFID protocol interoperability and supports basic reliability guarantees, provided by a 16-bit cyclic redundancy code (CRC16) and an on-chip 16-bit pseudo-random number generator (PRNG). The first version was published in 2004, and since then, there have been many attempts to secure EPCG2 protocols with the use of the passwords defined by the standard (e.g., [8,9]) or based on the CRC (e.g., [10,11]). Nevertheless, practically, they all have proven unsuccessful, due to the length of the keys, which are also static, and the linearity properties of CRC [12]. As a result, PRNG has become the key element in most security protocols proposed in the literature for this kind of tag (e.g., [13–17]). These protocols are based on the assumption that the PRNG implemented in the tag is cryptographically secure. In the new version (second) of the standard, tags may support one or more cryptographic suites (which must be specified), but then again, these would most likely require the implementation of a secure PRNG. The PRNG is

also used for some processes, such as the anti-collision algorithm or link-cover coding (a basic privacy mechanism described in the standard). Nevertheless, despite its practical significance, EPCG2 does not specify any possible PRNG implementation, and although security through obscurity has shown to be not advisable (e.g., [18,19]), manufacturers are still reluctant to make their designs publicly accessible. In addition, in the literature, there are only a few descriptions of PRNGs for low-cost RFID tags (e.g., [20–24]). Thus, as far as we know, the works of Melià *et al.* [25] and Mandal *et al.* [26], which is a modification of the previous one, are hitherto the only references that propose EPCG2 compliant PRNGs and that check how it meets the specific randomness requirements established by the standard.

Melià-Seguí *et al.* describe, in the first version [25] and then with more details in this journal [27], a PRNG for low-cost passive RFID tags (including, but not limited to EPCG2), called J3Gen, which provides a very high level of unpredictability, with a reduced computational complexity and low-power consumption. J3Gen is based on a linear feedback shift register (LFSR) configured with multiple feedback polynomials, and its authors claim that it is suitable for security purposes (e.g., [28,29]).

Nevertheless, in this paper we analyze the design of J3Gen and show that the security level provided by this PRNG falls well short of its security claims. Two different cases, for two different sets of suggested parameters, are analyzed. As a result, the randomness of the generated sequences decreases dramatically, and its use for security applications is questioned. We then suggest some values for the choice of parameters that could hinder these cryptanalyses, as well as some possible changes to strengthen the protocol. Note that the security of many cryptographic systems depends upon the generation of pseudo random numbers, and therefore, the weaknesses of the PRNG impact on the security of the global systems and may be exploited to attack them; e.g., the attack on Crypto-1, a cryptosystem for use on MIFARE chips [30].

The rest of the paper is organized as follows. Section 2 introduces some general concepts about EPCG2 PRNGs and describes the structure and the characteristics of J3Gen in particular. Section 3 describes the cryptanalysis of J3Gen for two different sets of recommended parameters. Then, in Section 4, we comment on some possible modifications to improve J3Gen, and finally, Section 5 concludes the paper.

2. An EPCG2-Compliant PRNG: J3Gen

Definition 1 (PRNG)**.** *A PRNG is a pseudo-random bit generator (PRG) whose output is partitioned into blocks of a given length,* n*. Each block defines an* n*-bit number, said to be drawn from the PRNG.*

Definition 2 (PRG)**.** *A PRG is a deterministic algorithm that, on inputting a binary string of length* K*, called the seed, generates a binary sequence,* s*, of length* $S >> K$ *which "appears" to be random.*

While it is very difficult to give a mathematical proof that a PRNG is indeed secure, we gain confidence by subjecting it to a variety of statistical tests designed to detect the specific characteristics expected of random sequences (we refer the reader to [31] for a comprehensive collection of randomness tests). Although the new version of EPCG2 explains that the different implemented cryptographic suites may define more stringent requirements for the PRNG [6], these are the "basic" randomness criteria set by the standard:

1. **Probability of a single** $RN16$: The probability that any $RN16$ drawn from the PRNG has value $RN16 = j$, for any j, shall be bounded by:

$$0.8/2^{16} < Prob(RN16 = j) < 1.25/2^{16}$$

2. **Probability of simultaneously identical sequences**: For a tag population of up to 10,000 tags, the probability that any two or more tags simultaneously generate the same sequence of $RN16s$ shall be less than 0.1%, regardless of when the tags are energized.

3. **Probability of predicting an** $RN16$: An $RN16$ drawn from a tag's PRNG shall not be predictable with a probability better than 0.025%, when the outcomes of prior draws from the PRNG under identical conditions are known.

According to the authors, J3Gen amply fulfills these requirements, providing a high level of security (an equivalent key size of 372 bits). We find, however, some flaws in the design of this PRNG that question the validity of this proposal. Before analyzing these issues, we describe the structure and the characteristics of J3Gen.

2.1. Description of J3Gen

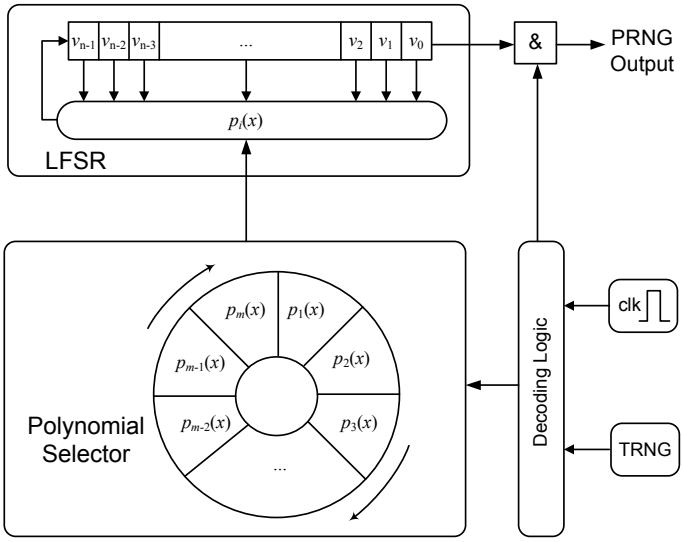
J3Gen is based on a dynamic linear feedback shift register (DLFSR) of n cells. A DLFSR can be defined, in turn, as an LFSR [32] where the feedback polynomial, $p_i(x)$, is not static, but changes dynamically [33]. J3Gen combines this DLFSR topology with a physical source of true randomness (thermal), which generates a "true random bit", denoted by trn. This bit controls the change of polynomials, preventing the linear behavior of the DLFSR. This trn is replaced by a PRNG in [26].

Figure 1 depicts the block diagram of J3Gen. A set of m primitive feedback polynomials are implemented as a wheel, and the polynomial selector rotates one position if $trn = 0$ and two positions (one position at one shift cycle and another at the next shift cycle) if $trn = 1$. These rotations are performed every l cycles, with $1 \leq l < n$. This value must be lower than n (the number of cells) to prevent a random number from being generated by a single feedback polynomial. The decoding logic is responsible for managing the internal PRNG clock and the trn bit, providing the correct signal to the different internal modules.

For a better understanding of the functioning of J3Gen, we review here the sample of the execution provided in [25]. The parameter configuration chosen for this example is: $n = 16$, $m = 8$ and $l = 15$. The value $n = 16$ for the LFSR size is selected because of compatibility reasons with EPCG2 (although larger values of n are also considered in [27]). The selected feedback polynomials ($m = 8$) should remain secret, since they can be considered as the secret key of the system. In [25], the LFSR states for 32 shift cycles are detailed, providing 32 output bits. Two true random values are used, which are set to $trn_1 = 0$ and $trn_2 = 1$. The system starts with $p_1(x)$ and outputs 15 ($= l$) bits until the TRNGmodule transfers $trn_1 = 0$ to the decoding logic module. Then, $p_2(x)$ is selected, and another 15 bits are generated, until the next (after l shift cycles) trn is obtained. As $trn_2 = 1$, the decoding logic rotates the polynomial selector one position at Shift 31 and another position at Shift 32. Eventually, two 16-bit pseudorandom numbers are generated; for the first one, $p_1(x)$ is used 15 cycles and $p_2(x)$ one cycle,

while for the second one, $p_2(x)$ is used 14 cycles and then $p_3(x)$ and $p_4(x)$ are used for one cycle each (p_4 will also be used 14 cycles for the next 16-bit pseudorandom number).

Figure 1. Block diagram of J3Gen. PRNG, pseudo-random number generator; LFSR, linear feedback shift register.



2.2. Security Strength

In [27], the authors equate the security strength of J3Gen with a key length, where each possible key corresponds to a possible feedback polynomial combination (which must be kept secret). Thus, for $n = 16$ and $m = 8$, this would mean a key size of roughly 73 bits; *i.e.*, 8 ($= m$) selected feedback polynomials out of 2,048 possible primitive polynomials of degree 16 ($= n$). Apart from the parameters, n and m, the polynomial update cycle, l, has also a major impact on the security level. For example, with $n = 32$ and $m = 16$, for $l = 31$, the authors compute that there will be up to four possible solutions for each system of equations, *i.e.*, up to four possible feedback polynomials could be involved in the generation of such a sequence. If $l = 25$, then the possible solutions are up to 16,384; for $l = 21$, the possible solutions increase up to 4,194,304, and so on, until $l = 1$, the extreme case, where all 67 million primitive feedback polynomials would be equally probable.

3. Cryptanalysis of the J3Gen

This section analyzes the security of J3Gen, describing two procedures that enable: (i) retrieving of the feedback polynomials; and (ii) reconstructing the sequence. These cryptanalyses have been carried out for $l = n - 1$ and $l = 1$, respectively. The former is the value selected for the execution sample in [25], and the latter is pointed out as the most secure option by the designers [27].

3.1. Case $l = n - 1$: Retrieving the Feedback Polynomials

This first analysis shows that the security evaluation carried out by the authors (see Section 2.2) presents some flaws. According to the given data, it can be inferred that the number of possible feedback

polynomials involved in the generation of a 16-bit pseudorandom number is estimated to be $2^{2(n-l)}$. Nevertheless, we show here that this number can be reduced dramatically to just 2^{n-l}. As a consequence, the case $l = n - 1$ becomes particularly vulnerable, and the feedback polynomials can be retrieved. This problem gets much worse when the adversary makes use of some known characteristics of the feedback polynomials.

3.1.1. Cryptanalysis Description

The knowledge of $2n$ output values of a sequence, s, generated with an n-degree feedback polynomial enables the definition of a linear equation system of n equations to retrieve such a polynomial:

$$O_{n\times 1} = S_{n\times n} \cdot C_{n\times 1} \Longrightarrow \begin{pmatrix} s_{n+1} \\ \vdots \\ s_{2n} \end{pmatrix} = \begin{pmatrix} s_n & \cdots & s_1 \\ \vdots & \ddots & \vdots \\ s_{2n-1} & \cdots & s_n \end{pmatrix} \cdot \begin{pmatrix} C_1 \\ \vdots \\ C_n \end{pmatrix} \quad (1)$$

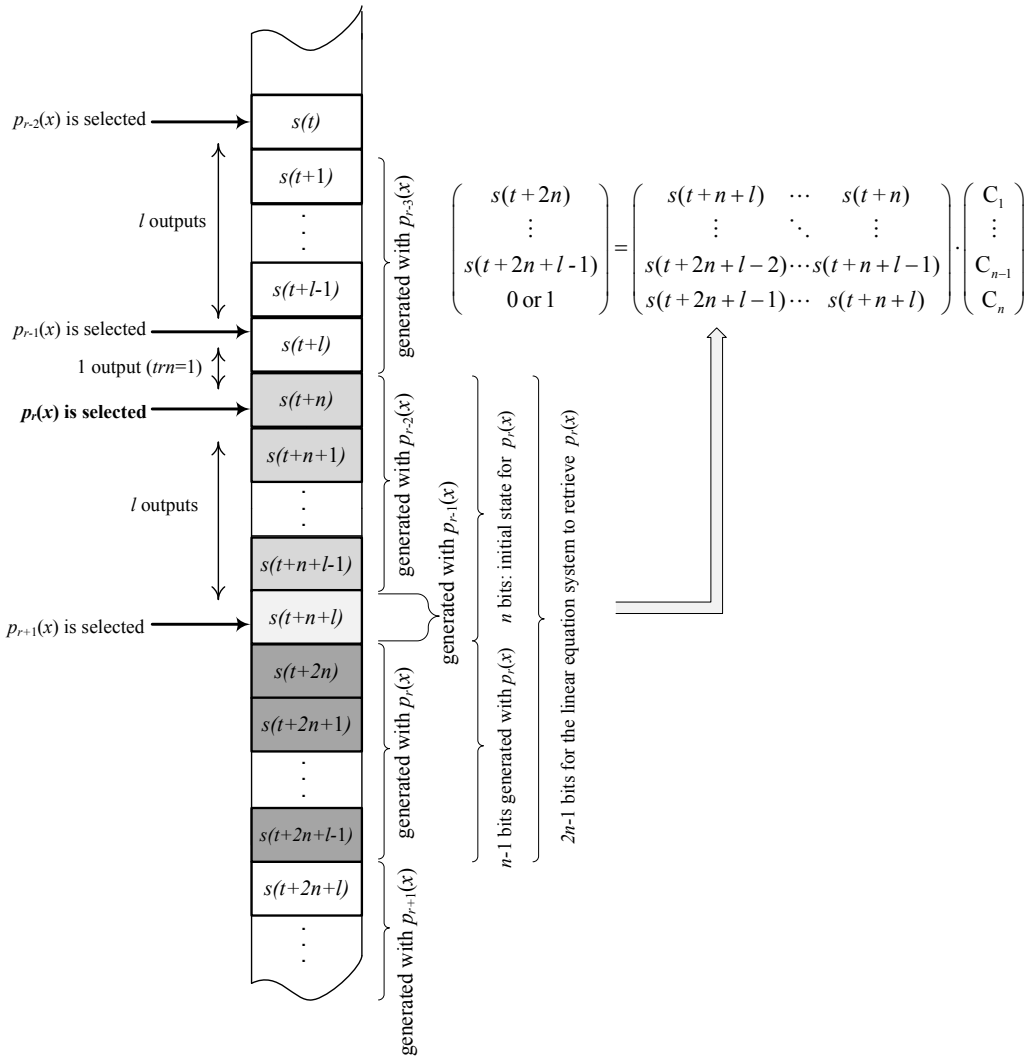
J3Gen prevents us from obtaining these values by changing the feedback polynomial after l rounds, with $l < n$. However, analyzing how DLFSR-based PRNGs work, it can be noticed that for $l = n - 1$, there is only one unknown bit left to define this linear equation system. Thus, an adversary just needs to try with the two possibilities, *i.e.*, zero and one, and checks if there exists a valid solution (only two feedback polynomials are involved). Figure 2 sketches, for a polynomial, $p_r(x)$, how the corresponding matrices are defined. When a feedback polynomial is selected by the polynomial selector, its initial state is known, as it corresponds with the following n outputs. If this feedback polynomial is used now to generate $l = n - 1$ outputs, we will then know a total of $2n - 1$ bits.

We have just shown that a feedback polynomial, $p_r(x)$, could be retrieved from $2n - 1$ outputs of J3Gen. However, if these $2n - 1$ bits are taken at random, with a high probability, fewer than $n - 1$ bits will have been generated with the same feedback polynomials, and the system will have no solution or this will be wrong. To overcome this problem, if more outputs of J3Gen are available, the adversary only has to shift one position after another and test if the defined system has a valid solution. If so, the polynomial is stored as a candidate. A maximum of n (if $trn = 1$) shifts will be needed before finding a correct solution. Finally, the adversary will have to pick up the correct feedback polynomials among the possible candidates. For this last task, different alternatives are possible, which will be commented on with an example in the next subsection. Given a J3Gen generated sequence, s, of length $S \geq n+l$ bits and letting s_k be the k-th bit of s, Algorithm 1 collects more formally the different steps of this cryptanalysis.

This general procedure can be further optimized when certain information about the feedback polynomials is known. For example, in J3Gen, the feedback polynomials are primitive. We know then that the number of non-zero terms is odd, with $C_0 = C_n = 1$. Thus, $n - 2$ equations, derived from just $2n - 2$ outputs, can be used to recover coefficients C_i with $i \in [1 : (n - 2)]$, while C_{n-1} will be the odd parity bit of such coefficients. This way, a submatrix $R = S_{(n-2)\times(n-2)}$ with the first $(n - 2)$ rows and columns of the previously defined $S_{n\times n}$ can be used to solve the linear equation system, while the $(n - 1)$-th and n-th rows of this matrix represent two spare equations that will only be included when required; that is, (i) when rank(R) = rank($R|O$) = $n - 4$ or (ii) when rank(R) = rank($R|O$) = $n - 3$. In the first case, we need to include both equations (($n - 1$) and n), while in the second case, we include

firstly the $(n-1)$-th row, and only if the rank does not change (*i.e.*,rank(R) = rank($R|O$) = $n-3$); the n-th row is included.

Figure 2. Sketch of the linear equation system definition.



Algorithm 1 Procedure to retrieve the J3Gen feedback polynomials.

Given a sequence, s, of length $S \geq n+l$

for $i = 1$ to $(S-(n+l)+1)$
Take $n+l$ bits of s: $h = s_i ... s_{i+n+l-1}$
Set $h_{2n} = j$ with $j = 0, 1$ and solve the linear equation systems.
If there exist a valid solution, then store it as a candidate.
end
Select the feedback polynomials among the candidates.

3.1.2. Cryptanalysis of the Given J3Gen Example

To illustrate the cryptanalysis, we apply it on the example described above (Section 2.1): $n = 16$, $m = 8$, $l = 15$ and the primitive feedback polynomials of Table 1. Figure 3 shows the results of the

cryptanalysis on a bitstring, s, of length $S = 176$ bits. There are 14 different candidates; seven of them correspond with actual used feedback polynomials, while the other six are "false positives". It is easy to note, however, that the frequency of appearance of genuine polynomials is much higher than that of "false positives"; when a candidate is found for a specific output, the probability of this being genuine is roughly 75%. Table 2 shows the average number of correct polynomials between the candidates for different lengths, S, and the number of them that are correct among the most frequent ones (chosen candidates). These results are obtained with the Monte-Carlo method for 1,000 repetitions. They show that 128 bits are enough to recover five of the eight feedback polynomials. These results can still be improved if the method to discard the "false positives" (*i.e.*, to choose among the candidates) is refined. For example, one could take into account the output when the candidate is obtained, as genuine solutions appear roughly every n outputs and/or if two solutions are found for the same output (Output 140 in Figure 3). In this case, only one of them can be correct, and therefore, the other can be ruled out directly.

Figure 3. Example of the results of the cryptanalysis.

	Output	Added rows		Coefficients (Taps)										
		15-th	16-th											
	33			1	3	4	5	6	7	11				
	34			1	3	4	5	6	7	11				
	35			1	3	4	5	6	7	11	**3rd Polynomial**			
	36	✓		1	3	4	5	6	7	11				
16	37			1	3	4	5	6	7	11				
	38	✓	✓	3	4	6	7	9	10	12	False Positive			
	38	✓	✓	1	3	4	5	6	7	11	**3rd Polynomial**			
	47			2	5	6	7	8	11	14	False Positive			
	48			2	5	6	7	8	11	14				
17	49			3	5	6	10	11	**4th Polynomial**					
	66			5	6	10	11	13	**6th Polynomial**					
16	67			5	6	10	11	13						
	81			1	2	3	4	5	6	12	False Positive			
	82			1	3	4	5	6	10	11				
16	83			1	3	4	5	6	10	11	**8th Polynomial**			
	84			1	3	4	5	6	10	11				
	98			4	5	6	7	11	**2nd Polynomial**					
14	99		✓	4	5	6	7	11						
	108			7	8	10	11	13	False Positive					
	109			7	8	10	11	13						
	112			1	3	4	5	6	7	11	**3rd Polynomial**			
15	113			1	3	4	5	6	7	11				
	127			5	6	11	**5th Polynomial**							
	128	✓		5	6	11								
13	129			5	6	11								
	130			5	6	11								
	140	✓	✓	4	5	6	10	11	**7th Polynomial**					
	140	✓	✓	4	5	6	7	9	12	13	14	15	False Positive	
	141	✓		4	5	6	10	11	**7th Polynomial**					
	142			4	5	6	10	11						
20	143	✓	✓	4	5	6	10	11						
	144	✓	✓	4	5	6	10	11						
	145	✓		4	5	6	10	11						
	146			4	5	6	10	11						
	160			1	3	4	5	6	10	11				
	161		✓	1	3	4	5	6	10	11				
16	162	✓		1	3	4	5	6	10	11	**8th Polynomial**			
	163			1	3	4	5	6	10	11				
	164			1	3	4	5	6	10	11				
	174			1	2	4	7	12	False Positive					
	176			4	5	6	7	11	**2nd Polynomial**					

Table 1. Primitive feedback 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 2. Average retrieved polynomials.

S	Correct Polynomials among the Candidates	$v/w = v$ Correct Polynomials among the w Most Frequent ones
31	0.2	0.2 / 1
32	0.23	0.2 / 1
48	1.3	1.2 / 2
64	2.4	2.1 / 3
80	3.4	3.2 / 4
96	4.4	3.7 / 5
112	5.3	4.6 / 6
128	5.7	5 / 7
144	6	5.5 / 8
160	5.9	5.4 / 8
176	6.7	6 / 8
192	7.2	6.5 / 8
208	7.5	6.5 / 8
224	7.5	6.6 / 8
240	7.6	6.6 / 8

Finally, note that this cryptanalysis does not require that the given outputs are consecutive. An adversary, with several outputs of length S^1, S^2,...S^n, can perform n independent analyses to retrieve one or several polynomials with each of them (provided that $S^i \geq l + n$).

3.1.3. Discussion for Different Values of the Parameters

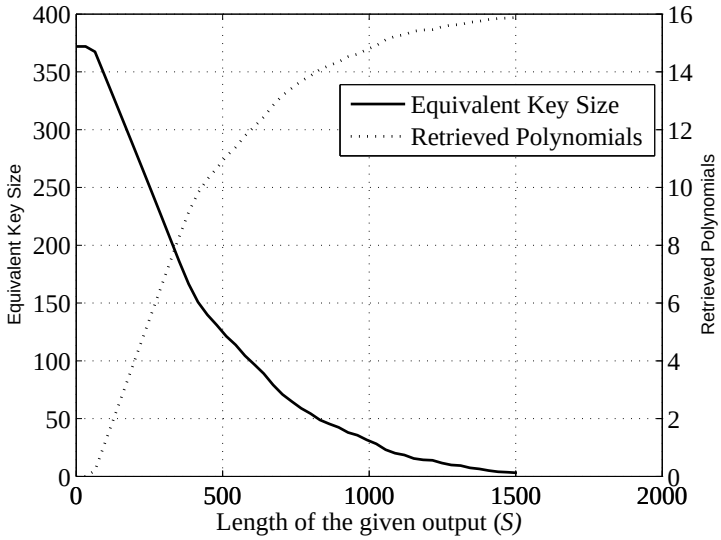
The cryptanalysis described above can be applied for any value of m. It only affects the number of outputs, S, that the cryptanalysis needs to recover all of the polynomials; $(n + m \cdot l)$ bits could be enough to recover up to m feedback polynomials. Each polynomial that is retrieved reduces the equivalent key size.

There is also no significant changes in the cryptanalysis when $n > 16$, other than the size of the linear equation systems and the number of shifts to define these. In [27], the authors opt for $m = 16$ and $n = 32$

as the best implementation in terms of security and hardware complexity. This configuration is supposed to provide an equivalent key of 372 bits (16 out of 67,108,864 primitive polynomials of degree 32). Nevertheless, 528 output bits could be enough to recover all of the 16 feedback polynomials, and we find that an adversary with an output sequence of 1,152 bits is able to recover all of these polynomials in half of the cases. Figure 4, based on simulations and statistical analysis, shows an approximation of the average number of polynomials that are retrieved for different lengths of the given outputs and how the equivalent key size reduces when these polynomials are disclosed.

The cases $l = n-2$ and $l = n-3$ with primitive feedback polynomials are essentially a particular case of the analysis described (with one or none of the spare equations). Apart from that, this cryptanalysis could be also applied to different values of l with a complexity significantly lower than that estimated by the authors [27]: 2^{n-l} possibilities instead of $2^{2(n-l)}$. However, this still leads to an ever-increasing computational complexity when $n - l$ increases, as the number of possibilities also does. Therefore, in this case, we suggest to consider other strategies. In the next section, for example, we describe a different approach to break J3Gen when $n - l$ is maximum ($l = 1$).

Figure 4. Results of the cryptanalysis for $m = 16$, $n = 32$ and $l = 31$.



3.2. Case $l = 1$: Reconstructing the Output Sequence

This section analyzes the J3Gen outputs for the case when $l = 1$, which is suggested as the most secure choice. This cryptanalysis is, as mentioned, different from the previous one and even more powerful. While the results in the previous case were probabilistic, we show here that the generated sequence can be reconstructed in a deterministic way. The analysis exploits a design fault in J3Gen that renders useless the randomness provided by trn; when $l = 1$, the m feedback polynomials are applied consecutively, *i.e.*, [$p_1(x)$, $p_2(x)$,..., $p_m(x)$, $p_1(x)$, $p_2(x)$,...]. Indeed, if $trn = 0$, the following output bit is computed by using the next feedback polynomial, $p_{i+1}(x)$, and then, another trn is obtained. If $trn = 1$, the system generates two output bits, by using $p_{i+1}(x)$ and $p_{i+2}(x)$, respectively, and then, another bit trn is drawn. Thus, the true random bit trn does no provide any randomness to the process, and each feedback polynomial will be periodically applied with period m. This fact makes

it possible to apply a cryptanalysis, such as that applied to programmable cellular automata (PCA) [34] and DLFSR [33], considering the output sequence, s, as an interleaved sequence composed by decimated sequences (*cf.* [35] for the theory on interleaved sequences).

3.2.1. Cryptanalysis Description

Let s be the output binary sequence produced by the J3Gen generator where the generic term $s(t) = v_0(t)$ is the least significant bit of the state $v(t) = (v_{n-1}(t), ..., v_1(t), v_0(t))$ of the LFSR at time instant t. The linear span of this random sequence can be defined as follows:

Definition 3 (Linear Span)**.** *The linear span (or linear complexity, notated as LC) of a binary sequence, s, is defined as the length of the shortest LFSR that can generate such a binary sequence.*

Let us also define the sequence, w_0, as a decimation of the sequence, s, by taking one term out of m; that is,

$$w_0(t) = s(t \cdot m) \text{ for } t \geq 0 \tag{2}$$

The following equation holds between the states of the LFSR,

$$v((t+1) \cdot m) = v(t \cdot m)]M \tag{3}$$

where:

$$M = \prod_{i=1}^{i=m} A_i \tag{4}$$

A_i being a $n \times n$ matrix, whose characteristic polynomial is the feedback polynomial, $p_i(x)$, of the LFSR. Thus, it can be written that:

$$w_0(t) = s(t \cdot m) = \pi(M^t \cdot v(0)) \tag{5}$$

where π is a linear map of an n-dimensional vector space over $GF(2)$ that transforms $(v_{n-1}(t), ..., v_1(t), v_0(t))$ into $v_0(t)$.

The term, $w_0(t+n)$, can be written [33,34] as a linear combination of the previous n terms, and consequently, the linear span of the sequence, w_0, is at most n. The same reasoning applies to any of the other decimated sequences, w_j, whose generic terms are defined as:

$$w_j(t) = s(t \cdot m + j), \ 0 \leq j \leq (m-1) \tag{6}$$

This way, the sequence, s, can be obtained by interleaving m different sequences w_j, where each of them has a linear span $LC \leq n$. Thus, the linear span of s is upper bounded as $LC(s) \leq n \cdot m$, which means that the sequence, s, can be reconstructed from the knowledge of at most $2n \cdot m$ bits. Consequently, for $n = 16$ and $m = 8$, the binary sequence produced by J3Gen can be reconstructed from just 256 consecutive bits (or 16 pseudorandom numbers), using an equivalent LFSR of 128 stages. In the case of $n = 32$ and $m = 16$, the output sequence can be rebuilt by using 1,024 consecutive bits (or 64 pseudorandom numbers). Note that such sequences can be reconstructed without the knowledge of the feedback polynomials.

These conclusions are confirmed when the Massey–Berlekamp algorithm [36] is applied on sequences generated by J3Gen with $n = 16$, $m = 8$ and $l = 1$ (the feedback polynomials of Table 1). The results

reveals a linear span $LC = 128$ with an equivalent feedback polynomial $p_{eq}(x) = x^{128} + x^{120} + x^{88} + x^{80} + x^{72} + x^{56} + 1$, whose order, which determines the period of the sequence, is $28,560$. Figure 5 and Figure 6 depict the linear span profile and the repetition period, respectively.

Figure 5. The linear span profile of the first 8,000 bits in a sequence of a length of 30,016 bits produced by the J3Gen PRNG with $n = 16$, $m = 8$ and $l = 15$.

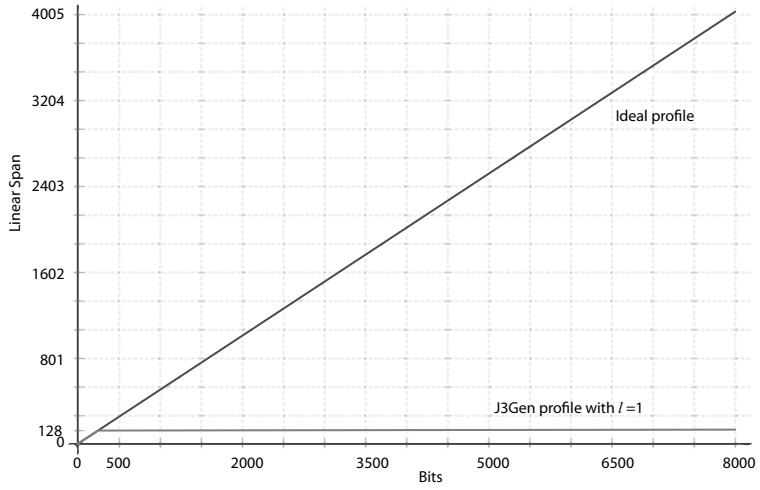
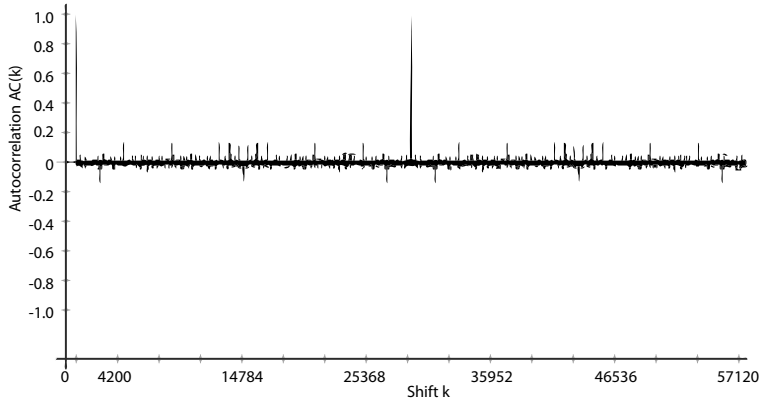


Figure 6. Autocorrelation of a sequence generated with $n = 16$, $m = 8$ and $l = 15$.



4. Security Recommendations for J3Gen

In [27], the authors suggest that the pool of feedback polynomials could include non-primitive polynomials to increase the number of possible combinations and, thus, prevent J3Gen from a brute force attack. This certainly would hinder the cryptanalysis described in Section 3.1, as the process to discard candidates (the last step in Algorithm 1) would become harder. However, this modification needs to be done carefully, since non-primitive polynomials produce sequences whose statistical properties are not guaranteed (must be proven). Furthermore, the selection of these feedback polynomials should not apply any fixed rule that could leak information about the selected protocols. For example, the polynomials in Table 1 seem to apply that used in similar architectures (e.g., [37]), which looks for an efficient hardware implementation by choosing polynomials with several coefficients in common: all polynomials share coefficients x^{16}, x^{11}, x^6, x^5 and x^0. This simplifies the logic circuitry (fewer gates)

to select the appropriated polynomials from the pool, but obviously impacts negatively on the global security of the PRNG.

Regarding the analysis described in Section 3.2, an alternative way to obtain the period of the sequence is with the computation of the matrix, M (see Equation (4)). This matrix can also be computed from $2n \cdot m$ consecutive bits taking only $2n$ bits of a decimated sequence (one out of m). The computation of M is equivalent to the computation of the equivalent LFSR determined by the linear span of a decimated sequence. Once the matrix is computed, its characteristic polynomial, $c(x)$, gives us information about the period of the sequence [33]. This matrix, M, is completely determined by the feedback polynomials and the order in which they are applied. Thus, it is possible to know *a priori* the period of the output binary sequence computing the characteristic polynomial of the matrix, M. The polynomials proposed in [25] (listed in Table 1) determine a matrix, M, whose characteristic polynomial is: $c(x) = x^{16} + x^9 + x^7 + x^6 + x^5 + x + 1 = (x^8 + x^7 + x^5 + x^3 + 1)(x^2 + x + 1)(x^3 + x + 1)^2$. The order of this non-primitive polynomial is $3,570$. Since $c(x)$ determines the period of all decimated sequences [33,35], the period of the interleaved sequence, s, is $3,570 \cdot m = 28,560$, a much lower value than the maximum length, $65,535$, produced by a single primitive LFSR of a length of 16. However, a simple modification in the order of the polynomials may increase the repetition period of the sequence. For example, if $p_1(x)$ and $p_5(x)$ interchange their positions, then the repetition period is maximized; the characteristic polynomial of the matrix, M, is $c(x) = x^{16} + x^{14} + x^{13} + x^{12} + x^{10} + x^8 + x^7 + x + 1$, which is primitive and determines a period of $65,536$ for the decimated sequences and $65,536 \cdot 8 = 524,288$ for the interleaved sequence. However, the pool of polynomials cannot significantly improve the upper bound of the linear span, since it does not depend on the specific implemented polynomials, but on their number and degree; $LC \leq n \cdot m$ (Section 3.2). Furthermore, the linear span will always be too low compared to the repetition period. As a consequence, we strongly advise against the use of $l = 1$.

5. Conclusions

In the present work, we have analyzed the security of J3Gen. J3Gen is one of the few PRNGs described in the literature that is suitable to be implemented in low-cost RFID tags and the only one, as far as we know, that has been shown to fulfill the randomness criteria established by the EPCglobal Gen2 standard. Nevertheless, despite its security claims, we have described here two distinct cryptanalytic attacks for the two values of the parameter, l, recommended by the designers:

- $l = n - 1$: A probabilistic cryptanalysis based on solving linear equation systems is introduced. This analysis allows one to recover the set of feedback polynomials, which constitute the secret information of J3Gen. No more than $(n + m \cdot l)$ output bits could be enough to accomplish this task. In addition, cases $l = n - 2$ and $l = n - 3$ are essentially straight derivations of the same analysis when the feedback polynomials are primitive.

- $l = 1$: A deterministic cryptanalysis based on the output sequence decimation is developed. This analysis shows that the entire output sequence of J3Gn can be reconstructed by the knowledge of $2\,nm$ bits of such a sequence.

Although these analyses undoubtedly question the security of J3Gen, we enumerated some recommendations that address these issues and could be helpful for this or future designs.

Acknowledgments

This work has been supported by the MICINNunder project "TUERI: Technologies for secure and efficient wireless networks within the Internet of Things with applications to transport and logistics", TIN2011-25452.

Author Contributions

No significant distinction can be made; the three co-authors have worked together and contributed equally to the reported research and writing of the paper.

Conflicts of Interest

The authors declare no conflicts of interest.

References

1. Ashton, K. That 'Internet of Things' Thing. *RFID J.* **2009**, *22*, 97–114.
2. Finkenzeller, K. *RFID Handbook : Fundamentals and Applications in Contactless Smart Cards and Identification*, 2nd ed.; John Wiley & Sons: Hoboken, NJ, USA, 2003.
3. European Comission. Commission launches consultation on Radio Frequency Identification (RFID). Available online: http://ec.europa.eu/digital-agenda/en/news/commission-launches-consultation-radio-frequency-identification-rfid (accessed on 4 April 2014).
4. The European Parliament and the Council of the European Union. Directive 95/46/EC.
5. Gohring, N. California Makes It a Crime to 'skim' RFID Tags. Available online: http://www.pcworld.com/article/151822/article.html (accessed on 4 April 2014).
6. EPC Global. UHF Air Interface Protocol Standard Generation2/Version2. Available online: http://www.gs1.org/gsmp/kc/epcglobal/uhfc1g2 (accessed on 4 April 2014).
7. ISO/IEC. Standard # 18000—RFID Air Interface Standard. Available online: http://www.hightechaid.com/standards/18000.htm (accessed on 4 April 2014).
8. Bailey, D.V.; Juels, A. Shoehorning security into the EPC standard. *Secur. Cryptogr. Netw.* **2006**, *4116*, 303–320.
9. Karthikeyan, S.; Nesterenko, M. RFID Security without Extensive Cryptography. In Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks, Alexandria, VA, USA, 7–10 November 2005; pp. 63–67.
10. Chien, H.Y.; Chen, C.H. Mutual authentication protocol for RFID conforming to EPC Class 1 Generation 2 standards. *Comput. Stand. Interfaces* **2007**, *29*, 254–259.
11. Qingling, C.; Yiju, Z.; Yonghua, W. A Minimalist Mutual Authentication Protocol for RFID System and BAN Logic Analysis. In Proceedings of ISECS International Colloquium on Computing, Communication, Control, and Management, Guangzhou, China, 3–4 August 2008.

12. Burmester, M.; Munilla, J. A Flyweight RFID Authentication Protocol. In Proceedings of the Workshop on RFID Security, Leuven, Belgium, 30 June–2 July 2009.
13. Choi, E.Y.; Lee, D.H.; Lim, J.I. Anti-cloning protocol suitable to EPCglobal Class-1 Generation-2 RFID systems. *Comput. Stand. Interfaces* **2009**, *31*, 1124–1130.
14. Burmester, M.; Munilla, J. Lightweight RFID authentication with forward and backward security. *ACM Trans. Inf. Syst. Secur.* **2011**, doi:10.1145/1952982.1952993.
15. Yoon, E. Improvement of the Securing RFID systems conforming to EPC Class 1 Generation 2 standard. *Expert Syst. Appl.* **2012**, *39*, 1589–1594.
16. Yeh, T.C.; Wang, Y.J.; Kuo, T.C.; Wang, S.S. Securing RFID systems conforming to EPC Class 1 Generation 2 standard. *Expert Syst. Appl.* **2010**, *37*, 7678–7683.
17. Huang, Y.J.; Yuan, C.C.; Chen, M.K.; Lin, W.C.; Teng, H.C. Hardware Implementation of RFID Mutual Authentication Protocol. *IEEE Trans. Ind. Electron.* **2010**, *57*, 1573–1582.
18. Engels, D.; Fan, X.; Gong, G.; Hu, H.; Smith, E.M. Hummingbird: Ultra-Lightweight Cryptography for Resource-Constrained Devices. *Financ. Cryptogr. Data Secur.* **2010**, *6054*, 3–18.
19. Bono, S.C.; Green, M.; Stubblefield, A.; Juels, A.; Rubin, A.D.; Szydlo, M. Security Analysis of a Cryptographically-Enabled RFID Device. In Proceedings of the 14th conference on USENIX Security Symposium, Baltimore, MD, USA, 31 July–5 August 2005.
20. Hell, M.; Johansson, T.; Meier, W. Grain: A stream cipher for constrained environments. *Int. J. Wirel. Mob. Comput.* **2007**, *2*, 86–93.
21. De Cannière, C.; Preneel, B. Trivium Specifications, ECRYPT Project. Available online: http://www.ecrypt.eu.org/stream/triviumpf.html (accessed on 4 April 2014).
22. Peris-López, P.; Hernández-Castro, J.; Estévez-Tapiador, J.; Ribagorda, A. LAMED–A PRNG for EPC Class-1 Generation-2 RFID specification. *Comput. Stand. Interfaces* **2009**, *31*, 88–97.
23. Martin, H.; San Millan, E.; Entrena, L.; Lopez, P.; Castro, J. AKARI-X: A pseudorandom number generator for secure lightweight systems. In Proceedings of the 2011 IEEE 17th International On-Line Testing Symposium (IOLTS), Athens, Greece, 13–15 July 2011; pp. 228–233.
24. Özcanhan, M.; Dalkilic, G.; Görle, M. An Ultra-Light PRNG for RFID Tags. In *Computer and Information Sciences III*; Gelenbe, E., Lent, R., Eds.; Springer: London, UK, 2013; pp. 231–238.
25. Melià-Seguí, J.; Garcia-Alfaro, J.; Herrera-Joancomartí, J. Multiple-polynomial LFSR based pseudorandom number generator for EPC Gen2 RFID tags. In Proceedings of the IECON 2011—37th Annual Conference on IEEE Industrial Electronics Society, Melbourne, Australia, 7–10 November 2011; pp. 3820–3825.
26. Mandal, K.; Fan, X.; Gong, G. Warbler: A Lightweight Pseudorandom Number Generator for EPC C1 Gen2 Passive RFID Tags. *Int. J. RFID Secur. Cryptogr.* **2013**, *2*, 82–91.
27. Melià-Seguí, J.; Garcia-Alfaro, J.; Herrera-Joancomartí, J. J3Gen: A PRNG for Low-Cost Passive RFID. *Sensors* **2013**, *13*, 3816–3830.
28. Delgado-Mohatar, O.; Fúster-Sabater, A.; Sierra, J.M. A light-weight authentication scheme for wireless sensor networks. *Ad Hoc Netw.* **2011**, *9*, 727–735.
29. Dolev, S.; Gilboa, N.; Kopeetsky, M.; Persiano, G.; Spirakis, P. Information security for sensors by overwhelming random sequences and permutations. *Ad Hoc Netw.* **2014**, *12*, 193–200.

30. Nohl, K.; Evans, D.; Starbug, S.; Plötz, H. Reverse-engineering a Cryptographic RFID Tag. In Proceedings of the 17th Conference on Security Symposium; San Jose, CA, USA, 28 July–1 August 2008; pp. 185–193.
31. Rukhin, A.; Soto, J.; Nechvatal, J.; Smid, M.; Barker, E.; Leigh, S.; Levenson, M.; Vangel, M.; Banks, D.; Heckert, A.; *et al.* A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. Available online: http://csrc.nist.gov/publications/nistpubs/800-22-rev1a/SP800-22rev1a.pdf (accessed on 4 April 2014).
32. Golomb, S. *Shift-Register Sequences*; Aegean Park Press: Laguna Hill, CA, USA, 1982.
33. Peinado, A.; Fúster-Sabater, A. Generation of pseudorandom binary sequences by means of linear feedback shift registers (LFSRs) with dynamic feedback. *Math. Comput. Model.* **2013**, *57*, 2596–2604.
34. Blackburn, S.R.; Murphy, S.; Paterson, K.G. Comments on "Theory and Applications of Cellular Automata in Cryptography". *IEEE Trans. Comput.* **1997**, *46*, 637–639.
35. Gong, G. Theory and applications of q-ary interleaved sequences. *IEEE Trans. Inf. Theory* **1995**, *41*, 400–411.
36. Massey, J. Shift-register synthesis and BCH decoding. *IEEE Trans. Inf. Theory* **1969**, *15*, 122–127.
37. Mita, R.; Palumbo, G.; Pennisi, S.; Poli, M. Pseudorandom bit generator based on dynamic linear feedback topology. *Electron. Lett.* **2002**, *38*, 1097–1098.