EPCGen2 Pseudorandom Number Generators: Analysis of J3Gen

A. Peinado[a], J. Munilla[a,*], A. Fúster-Sabater[b]

[a]*E.T.S.I. Telecomunicación, Universidad de Málaga, Málaga (Spain)*
[b]*Instituto de Tecnologías Físicas y de la Información, CSIC, Madrid (Spain)*

Abstract

This paper analyzes the cryptographic security of J3Gen, a promising pseudo random number generator for low-cost passive 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 which question its security claims: *i*) a probabilistic attack based on solving linear equation systems, and *ii*) a deterministic attack based on the output sequence decimation. 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 crytanalyze J3Gen. We then make some recommendations which address these issues.

Keywords: PRNG, security, cryptanalytic attack, RFID, EPCglobal Gen2

1. Introduction

The term "Internet of Things" (IoT) was coined in 1999 by Kevin Ashton, one of the cofounder of Auto-ID [1]; a center that promoted research and development of tracking products for the supply-chain by using low-cost RFID tags. The advantages of RFID over barcode technology are that it is wireless, does not require direct line-of-sight, and tags can be interrogated at greater distances, faster and concurrently [2]. This makes the IoT

*Corresponding author.
E-mail address: munilla@ic.uma.es
Phone: (+34) 952134166

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 managements and automatic identification in general. 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 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.

Initial designs of RFID protocols focused on 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 CASPIAN (Consumer Against Supermarket Privacy Invasion and Numbering) complained against the possible misuse of the 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 by the use of RFID technology, particularly in terms of privacy, data protection and information security [3]. RFID can be indeed used to perform different privacy invasions, such as unauthorized reading or tracking 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 US), the technical mean to control access to tags is the implementation of cryptographical mechanisms which take into account their special characteristics: power-constrained devices, 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 Specificiation [6]. EPCglobal Gen2, hereafter EPCG2, is the standard (ISO [7]) for low-cost tags which work in the UHF band 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]). However, practically they all, due to the length of the keys, which are also static, and the linearity properties of CRC, have proven unsuccessful [12]. As a result, PRNG has become the key element in most security protocols proposed in the literature for this kind of tags (e.g. [13, 14, 15, 16, 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 process such as the anti-collision algorithm or the link-cover coding (a basic privacy mechanism described in the standard). However, despite its practical significance, EPCG2 does not specify any possible implementation, manufacturers are reluctant to make publicly accessible their PRGN designs, and in the literature there are only a few descriptions of PRNGs for low cost RFID tags (e.g., [18, 19, 20]). Thus, as far as we know, the work of Melià *et al.* [21] is hitherto the only reference which proposes an EPCG2 compliant PRNG and checks how it meets the specific randomness requirements established by the standard.

Melià-Seguí *et al.* describe, in a first version [21] and then with more details [22], 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. [23, 24]). However, 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 cryptanalyzed. 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 which could hinder these cryptanalyses, as well as some possible changes to strengthen the protocol.

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 cryptanalyzes J3Gen for two different sets of recommended parameters. Then, in Section 4, we comment 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 a n-bit number, said to be drawn from the PRNG.*

Definition 2 (PRG). *A PRG is a deterministic algorithm that, on input 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 [25] for a comprehensive collection of randomness tests). Although the new version of the standard [6] explains that the different implemented cryptographic suites may define more stringent requirements for the PRNG, these are the "basic" randomness criteria set by EPCG2:

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 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 (equivalent key size of 372 bits). We find, however, some flaws in the design of this PRNG, which 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 a LFSR [26] where the feedback polynomial, $p_i(x)$, is not static but changes dynamically [27]. 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.

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 (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 execution provided in [21]. 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 due to compatibility reasons with EPCG2 (although larger values of n are also considered in [22]). The selected feedback polynomials ($m = 8$) should remain secret as they can be considered as the secret key of the system. In [21], 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 $l = 15$ bits until the TRNG module transfers $trn_1 = 0$ to the Decoding Logic module. Then, $p_2(x)$ is selected, and another $l = 15$ 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 $p_2(x)$ 1 cycle, while for the second one,

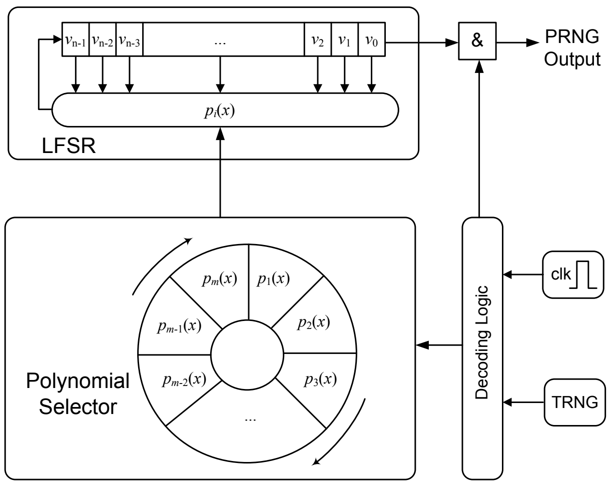


Figure 1: Block Diagram of J3Gen.

$p_2(x)$ is used 14 cycles, and then $p_3(x)$ and $p_4(x)$ are used for 1 cycle each (p_4 will be also used 14 cycles for the next 16-bit pseudorandom number).

2.2. Security Strength

In [22] the authors equates 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 $m = 8$, it 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 4 possible solutions for each system of equations, *i.e.* up to 4 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 which enable: *i*) to retrieve the feedback polynomials, and *ii*) to reconstruct 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 [21], and the latter is pointed out as the most secure option by the designers [22].

3.1. Case $l = n - 1$: retrieving the feedback polynomials

This first analysis shows that the security evaluation carried out by the authors (see Sec. 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 $2^{2(n-l)}$. However, 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 known characteristics of the feedback polynomials.

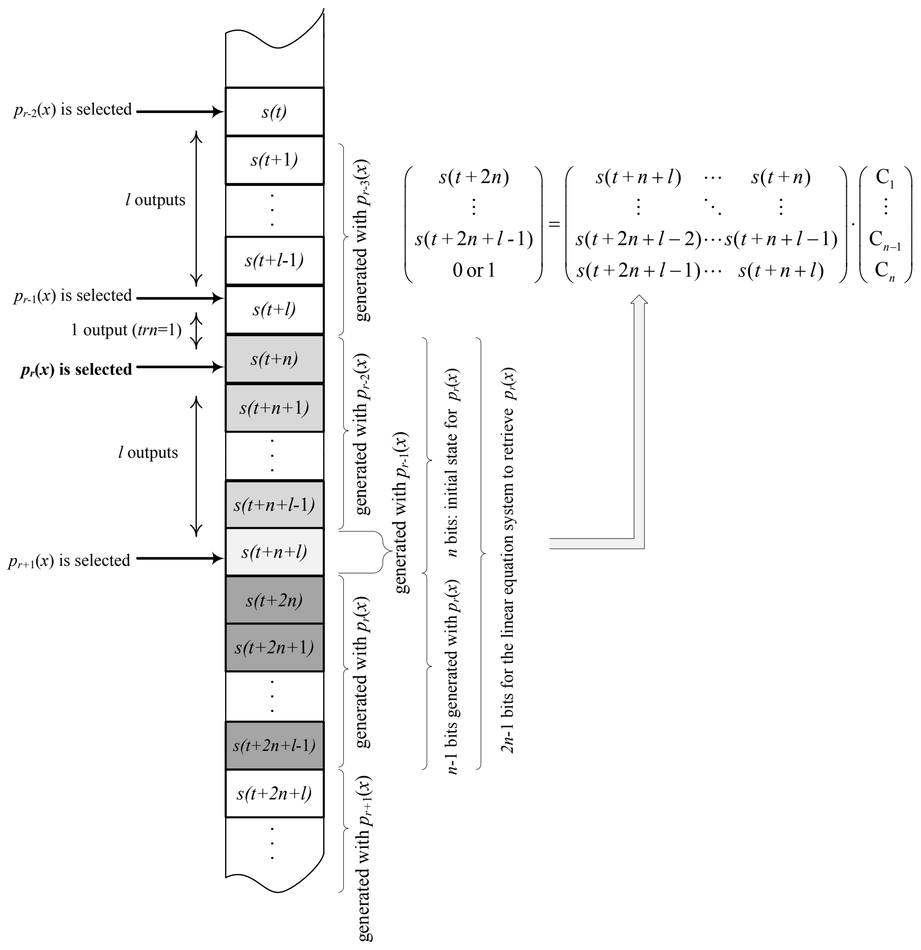
3.1.1. Cryptanalysis Description

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

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

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.* 0 and 1, 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



$$\begin{pmatrix} s(t+2n) \\ \vdots \\ s(t+2n+l-1) \\ 0 \text{ or } 1 \end{pmatrix} = \begin{pmatrix} s(t+n+l) & \cdots & s(t+n) \\ \vdots & \ddots & \vdots \\ s(t+2n+l-2) & \cdots & s(t+n+l-1) \\ s(t+2n+l-1) & \cdots & s(t+n+l) \end{pmatrix} \cdot \begin{pmatrix} C_1 \\ \vdots \\ C_{n-1} \\ C_n \end{pmatrix}$$

Figure 2: Sketch of the linear equation system definition.

```
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.
```

Figure 3: Procedure to retrieve the J3Gen Feedback Polynomials

random, with 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 with an example in the next subsection. Given a J3Gen generated sequence s of length $S \geq n+l$ bits, and let s_k be the k-th bit of s, Figure 3 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 liner equation system, while the $(n-1)$-th and n-th rows of this matrix represent two spare equations that will be only 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.

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 4 shows the results of the cryptanalysis on a bitstring s of length $S = 176$ bits. There are 14 different candidates; 7 of them correspond with actual used feedback polynomials, while the other 6 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 an 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 which are correct among the most frequent ones (chosen candidates). These results are obtained with the Monte-Carlo method for 1000 repetitions. They show that 128 bits are enough to recover 5 of the 8 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 4). In this case, only one of them can be correct and therefore the other can be ruled out directly.

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

Table 1: Primitive Feedback Polynomials

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

	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					
15	112			1	3	4	5	6	7	11			**3rd Polynomial**
	113			1	3	4	5	6	7	11			
	127			5	6	11							
13	128	✓		5	6	11							**5th Polynomial**
	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					
	142			4	5	6	10	11					
20	143	✓	✓	4	5	6	10	11					**7th Polynomial**
	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**

Figure 4: Example of results of the cryptanalysis.

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

3.1.3. Discussion for different values of the parameters

The cryptanalysis 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 [22], 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 1152 bits is able to recover all of these polynomials in the half of the cases. Figure 5, based on simulations and statistical analysis, shows an approximation of the average number of polynomials that are retrieved for different lengths of given outputs, and how the equivalent key size reduces when these polynomials are disclosed.

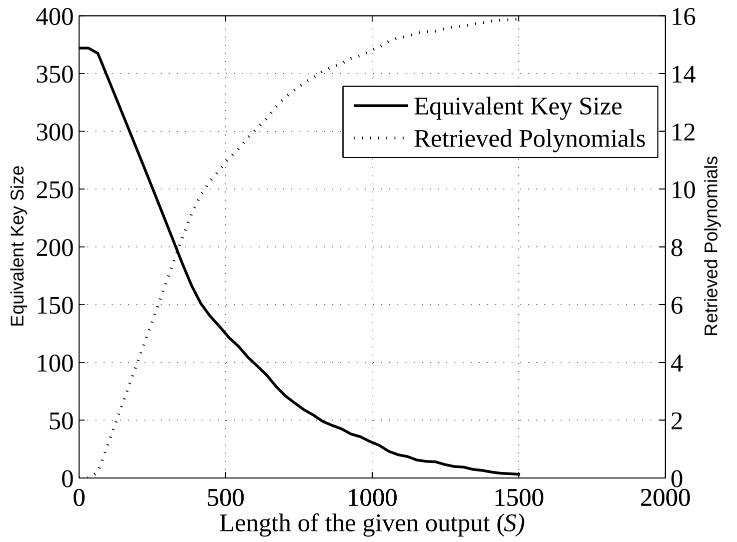


Figure 5: Cryptanalysis results for $m = 16$, $n = 32$ and $l = 31$.

Apart from the cases $l = n - 2$ and $l = n - 3$ with primitive feedback polynomials, which are essentially a particular case of the analysis described above (with one or none of the spare equations), this cryptanalysis could be also adapted to different values of l with a complexity significantly lower than that estimated by the authors [22]: 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 cryptanalyze the case when $n-l$ is maximum ($l = 1$).

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 analysis, as just mentioned, is different to the previous one and much 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) [28] and DLFSR [27], considering the output sequence s as an interleaved sequence composed by decimated sequences (*cf.* [29] for 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 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 a 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 [27, 28] 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:

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

In 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 [30] is applied on sequences generated by J3Gen with $n = 16$, $m = 8$ and $l = 1$ (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 6 and Figure 7 depict the linear span profile and the repetition period respectively.

4. Security Recommendations for J3Gen

In [22] 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 (last step in Figure 3) 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 proved). Furthermore, the selection of these feedback polynomials should not apply any fix rule which could leak information about the selected protocols. For example, polynomials in Table 1 seem to apply that used in similar architectures (e.g. [31]) which looks for efficient hardware implementation by

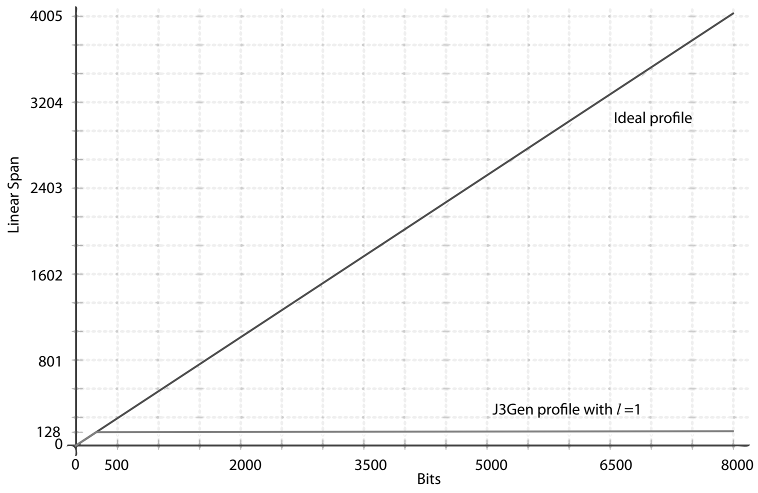


Figure 6: Linear span profile of the first 8,000 bits in a sequence of length 30,016 bits produced by the J3Gen PRNG with $n = 16$, $m = 8$ and $l = 15$.

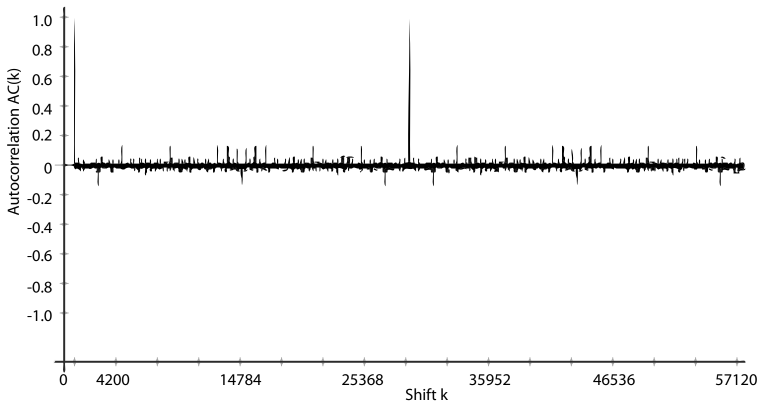


Figure 7: Autocorrelation of a sequence generated with $n = 16$, $m = 8$ and $l = 15$.

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 [27]. 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 [21] (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 [27, 29], 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 length 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 significatively improves the upper bound of the linear span, $LC \leq n \cdot m$ (Section 3.2), since it does not depend on the specific implemented polynomials but on their number and degree. Furthermore, the linear span always will 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 which is suitable to be implemented on low-cost RFID tags, and the only one, as far as we know, that has shown to fulfill the randomness criteria established by the EPCglobal Gen2 standard. However, despite its security claims, we have described here two distinct crypanalytic 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 $2nm$ bit of such sequence.

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

Acknowledgments

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

References

[1] K. Ashton, That 'Internet of Things' Thing, RFID Journal.
URL `http://www.rfidjournal.com/articles/view?4986`

[2] K. Finkenzeller, RFID Handbook : Fundamentals and Applications in Contactless Smart Cards and Identification, 2nd Edition, John Wiley & Sons, 2003.

[3] European Comission, Commission launches consultation on Radio Frequency Identification (RFID). http://ec.europa.eu/digital-agenda/en/news/commission-launches-consultation-radio-frequency-identification-rfid.

[4] The European Parliament and the Council of the European Union., Directive 95/46/EC.

[5] N. Gohring, California Makes It a Crime to 'skim' RFID Tags, PC-World (october 2008).

[6] EPC Global, UHF Air Interface Protocol Standard Generation2/Version2,
http://www.gs1.org/gsmp/kc/epcglobal/uhfc1g2.

[7] ISO/IEC, Standard # 18000 – RFID Air Interface Standard,
http://www.hightechaid.com/standards/18000.htm.

[8] D. V. Bailey, A. Juels, Shoehorning security into the epc standard, in: In International Conference on Security in Communication Networks SCN 2006, volume 4116 of LNCS, Springer-Verlag, 2006, pp. 303–320.

[9] S. Karthikeyan, M. Nesterenko, RFID Security without Extensive Cryptography, in: Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks, SASN '05, ACM, New York, NY, USA, 2005, pp. 63–67. doi:10.1145/1102219.1102229.
URL `http://doi.acm.org/10.1145/1102219.1102229`

[10] H.-Y. Chien, C.-H. Chen, Mutual authentication protocol for RFID conforming to EPC Class 1 Generation 2 standards, Comput. Stand. & Interfaces 29 (2) (2007) 254–259. doi:http://dx.doi.org/10.1016/j.csi.2006.04.004.

[11] C. Qingling, Z. Yiju, W. Yonghua, A Minimalist Mutual Authentication Protocol for RFID System and BAN Logic Analysis, Computing, Communication, Control and Management, ISECS International Colloquium on 2 (2008) 449–453. doi:http://doi.ieeecomputersociety.org/10.1109/CCCM.2008.305.

[12] M. Burmester, J. Munilla, A Flyweight RFID Authentication Protocol, in: Workshop on RFID Security – RFIDSec'09, Leuven, Belgium, 2009.

[13] E. Y. Choi, D. H. Lee, J. I. Lim, Anti-cloning protocol suitable to EPCglobal Class-1 Generation-2 RFID systems, Comput. Stand. Interfaces 31 (6) (2009) 1124–1130. doi:http://dx.doi.org/10.1016/j.csi.2008.12.002.

[14] M. Burmester, J. Munilla, Lightweight RFID authentication With Forward and Backward Security, ACM Trans. Inf. Syst. Secur. 14 (1) (2011) 11.

[15] E. Yoon, Improvement of the Securing RFID systems conforming to EPC Class 1 Generation 2 standard, Expert Systems with Applications 39 (2012) 1589–1594.

[16] T.-C. Yeh, Y.-J. Wang, T.-C. Kuo, S.-S. Wang, Securing RFID systems conforming to EPC Class 1 Generation 2 standard, Expert Syst. Appl. 37 (12) (2010) 7678–7683. doi:10.1016/j.eswa.2010.04.074.
URL `http://dx.doi.org/10.1016/j.eswa.2010.04.074`

[17] Y.-J. Huang, C.-C. Yuan, M.-K. Chen, W.-C. Lin, H.-C. Teng, Hardware Implementation of RFID Mutual Authentication Protocol, IEEE Transactions on Industrial Electronics 57 (5) (2010) 1573–1582.

[18] M. Hell, T. Johansson, W. Meier, Grain: a stream cipher for constrained environments, Int. J. Wire. Mob. Comput. 2 (1) (2007) 86–93. doi:10.1504/IJWMC.2007.013798.
URL `http://dx.doi.org/10.1504/IJWMC.2007.013798`

[19] C. de Cannière, B. Preneel, Trivium Specifications; Technical Report; ECRYPT Project., http://www.ecrypt.eu.org/stream/triviumpf.html (2008).

[20] P. Peris-López, J. Hernández-Castro, J. Estévez-Tapiador, A. Ribagorda, LAMED: A PRNG for EPC Class-1 Generation-2 RFID specification, Computer Standards & Interfaces 31 (1) (2009) 88–97. doi:10.1016/j.csi.2007.11.013.

[21] J. Melià-Seguí, J. Garcia-Alfaro, J. Herrera-Joancomartí, Multiple-polynomial LFSR based pseudorandom number generator for EPC Gen2 RFID tags, in: IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society, 2011, pp. 3820–3825. doi:10.1109/IECON.2011.6119932.

[22] J. Melià-Seguí, J. Garcia-Alfaro, J. Herrera-Joancomartí, J3Gen: A PRNG for Low-Cost Passive RFID, Sensors 13 (3) (2013) 3816–3830. doi:10.3390/s130303816.
URL `http://www.mdpi.com/1424-8220/13/3/3816`

[23] O. Delgado-Mohatar, A. Fúster-Sabater, J. M. Sierra, A light-weight authentication scheme for wireless sensor networks, Ad Hoc Networks 9 (5) (2011) 727–735. doi:10.1016/j.adhoc.2010.08.020.
URL `http://dx.doi.org/10.1016/j.adhoc.2010.08.020`

[24] S. Dolev, N. Gilboa, M. Kopeetsky, G. Persiano, P. Spirakis, Information security for sensors by overwhelming random sequences and permutations, Ad Hoc Networks (In Press, Corrected Proof)doi:10.1016/j.adhoc.2011.09.002.
URL `http://dx.doi.org/10.1016/j.adhoc.2011.09.002`

[25] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo, A statistical test suite for random and pseudorandom number generators for cryptographic applications, http://csrc.nist.gov/rng/, Technical Report (2001).

[26] S. Golomb, Shift-Register Sequences, Aegean Park Press, Laguna Hill, California, 1982.

[27] A. Peinado, A. Fúster-Sabater, Generation of pseudorandom binary sequences by means of linear feedback shift registers (LFSRs) with dynamic feedback, Mathematical and Computer Modelling 57 (1112) (2013) 2596 – 2604. doi:http://dx.doi.org/10.1016/j.mcm.2011.07.023.

[28] S. R. Blackburn, S. Murphy, K. G. Paterson, Comments on "Theory and Applications of Cellular Automata in Cryptography", IEEE Trans. Comput. 46 (5) (1997) 637–638. doi:10.1109/12.589245.
URL `http://dx.doi.org/10.1109/12.589245`

[29] G. Gong, Theory and applications of q-ary interleaved sequences, Information Theory, IEEE Transactions on 41 (2) (1995) 400–411. doi:10.1109/18.370141.

[30] J. Massey, Shift-register synthesis and BCH decoding, IEEE Transactions on Information Theory 15 (1) (1969) 122–127. doi:10.1109/tit.1969.1054260.
URL `http://dx.doi.org/10.1109/tit.1969.1054260`

[31] R. Mita, G. Palumbo, S. Pennisi, M. Poli, Pseudorandom bit generator based on dynamic linear feedback topology, Electronics Letters 38 (19) (2002) 1097–1098. doi:10.1049/el:20020750.