Improved Lightweight Pseudo-Random Number Generators for the Low-Cost RFID Tags

Jiageng Chen*, Atsuko Miyaji†, Hiroyuki Sato‡ and Chunhua Su§
*†‡§ School of Information Science,
Japan Advanced Institute of Science and Technology,
Asahidai 1–1, Nomi-shi, Ishikawa, 923–1292 Japan
Email: *jg-chen@jaist.ac.jp, †miyaji@jaist.ac.jp, ‡s1310028@jaist.ac.jp, §chsu@jaist.ac.jp
†CREST, Japan Science and Technology Agency,
Kawaguchi Center Building 4–1–8, Honcho, Kawaguchi-shi,
Saitama, 332–0012 Japan

***Abstract*—EPC Gen2 tags are working as international RFID standards for the use in the supply chain worldwide, such tags are computationally weak devices and unable to perform even basic symmetric-key cryptographic operations. For this reason, to implement robust and secure pseudo-random number generators (PRNG) is a challenging issue for low-cost Radio-frequency identification (RFID) tags. In this paper, we study the security of LFSR-based PRNG implemented on EPC Gen2 tags and exploit LFSR-based PRNG to provide a better constructions. We provide a cryptanalysis against the J3Gen which is LFSR-based PRNG and proposed by Sugei *et al.* [1], [2] for EPC Gen2 tags using distinguish attack and make observations on its input using NIST randomness test. We also test the PRNG in EPC Gen2 RFID Tags by using the NIST SP800-22. As a counter-measure, we propose two modified models based on the security analysis results. We show that our results perform better than J3Gen in terms of computational and statistical property.**

***Keywords*—lightweight PRNG, EPC Gen2 RFID tag, randomness test**

I. Introduction

Pseudo-random number generator is one of the major security components in RFID System, especially for EPC Gen2 low-cost RFID [3] whose security mainly reply on its security. As deployed by EPCglobal, the RFID tags are passive and carry what is known as Electronic Product Code (EPC). They are usually passive and capable of transmitting a static identifier or serial number for a short distance. Typically, the tags are activated by a query from a nearby reader, which also transmits power for the operation of the tag. Due to the power and hardware limitations, the common security tools such as hash function and data encryption are too expensive. Most authentication protocol proposals involve a challenge response mechanism between the reader and the tag, as this scheme is well-known, efficient, easy to implement and provides with adequate security for most applications.

In a RFID authentication session, reader sends a challenge to a tag, and the tag must reply with a valid response to the reader in order to be authenticated [4]. Authentication protocols, even for low-cost EPC Gen2 ones, should be resilient against attacks based on eavesdropping multiple challenge-response pairs [5]. This is the reason why cryptographic solutions propose mutual authentication protocols where both reader and tag must convince each other that they both know a shared secret. One way for this to be done is by including nonces (random numbers only used once) in the challenge-response exchanges. The security of PRNG plays an extremely important roles in EPC Gen2 low-cost RFID authentication.

This paper aims to study the problem that whether we can trust the randomness which are generated by the low-cost EPC Gen2 RFID. Melia-Segui proposed multiple-polynomial LFSR based PRNG for EPC Gen2 RFID Tags in 2011 [1]. It is configured with multiple feedback polynomials and is based on a linear feedback shift register (LFSR) which can be efficiently implemented on hardware and satisfy the randomness requirements of EPC Gen2 standard with a simple design. In this paper, we focus on the security on multiple-polynomial LFSR based PRNG and investigate the properties of such lightweight constructions.

A. related works

There are many proposals for random number generation in RFID tags. Generally, there are two main approaches for constructing such mechanisms to attain random number inside RFID tags. The first approach is based on a physical source such as thermal noise of zener diodes or radioactive decay can be used to generate truly random numbers. However, this method bare a common drawback of the truly random number generator, which is its inefficiency of aggregating many physical resource and the impossibility of replicating their outputs, so it is not practical. The other approaches Pseudorandom Number Generators (PRNGs), which can be artificially generated inside the tag using some mathematical methods such as Linear Congruential Generator (LCG) and Linear Feedback Shift Register (LFSR) [6].

For the PRNGs for low-cost RFID tags, a lightweight construction was proposed by Mandal *et al.* in 2011 [7]. It is NLFSR (Non-Linear Feedback Shift Register) based PRNG. It requires 36 clock cycles for key initialization and 80 clock cycles for running phase. It can be used for securing tag identification protocols and suitable for EPC Class 1 Generation 2. Martin, H. *et al.* proposed AKARI in 2011 [8], it has two variants of AKARI , AKARI-1 and AKARI-2. Their proposal improves the reliability and security of the system. Wu *et al.* proposed ultra- lightweight true random number generators (TRNGs) in 2010 [9]. These are based on the concept that

the resulting state may be random, when a circuit switches from a metastable state to a bi-stable state. It is lightweight TRNGs with low hardware cost. Peris *et al.* proposed LAMED in 2009 [10] which is a realistic approach for low cost RFID tags. The output of LAMED succeeded in all randomness tests and can be implemented with less number of gates can be easily implemented in hardware.

Che *et al.* presented a new PRNG [11] 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. But, Joan *et al.* proved that the scheme does not achieve the handling the linearity of LFSRs. They showed how an eavesdropper may obtain the feedback polynomial of the LFSR by using very few observations. So, this PRNG can be obtained the internal state $(n = 16)$ by using very few observations.

$$P_{success}(3n - 1) = 0.1328 \quad (1)$$

B. our contributions and paper organization

In this paper, we revisit the current trends of design the PRNG for EPC Gen2 tags and propose a distinguish attack the multiple polynomial LFSR based PRNG J3Gen. We argue that some recent results of the cryptanalysis against J3Gen may not work properly as it was claimed in [12]. We provide a more detailed cryptanalysis for multiple polynomial selection PRNGs in [1], [2]. Based on our cryptanalysis, we found that the previous works can be easily attack by distinguish attack. From the randomness test we propose two variants of multiple polynomial selection PRNGs. We provide both experimental and theoretical analysis and show our improved constructions which can perform better in security aspect and with less biased output.

The rest of the paper is organized as follows. In Section 2, we provide some brief introductions about the preliminary to be used in our analysis and proposal. We present our cryptanalysis against J3Gen LFSR-based PRNG in Section 3, we show that our distinguish attack can easy distinguish the output from a true random sequence . Section 4 proposes ours PRNG (two modified schemes). Section 5 describes experimental results and evaluations. In Section 6, we draw the conclusions about our research on LFSR-based PRNG for EPC Gen2 tags.

II. Preliminaries

In this section, we briefly introduce some primitives which are used to construct pseudo-random number generand evaluate the quality in our paper.

A. Linear feedback shift registers

Linear Feedback Shift Registers (LFSR) are linear registers generating bits on their iterations of the internal states. Each bit of LFSR can be either a shifted bit from the internal state or the result of a simple algebraic computation from internal state. The bit positions that affect the next state are called the taps and tap positions run from the output of one register within the LFSR into XOR gates that determine input to another register within the LFSR. These are chosen based on the primitive polynomial their feedback can be expressed in finite field arithmetic as a polynomial. 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. 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.

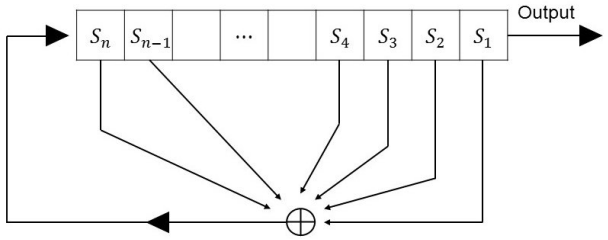

Figure 4 LFSR

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. For example, let $s_{k+1}, s_{k+2}, ..., s_{k+2n}$ be a sequence of $2n$ consecutive bits generated from an LFSR. Let $c_n, c_{n-1}, ..., c_1$ be the feedback function of the LFSR. Then, the feedback function can be easily computed by solving the following equation system,

By solving this equation, we can obtain the feedback polynomial coefficients. Therefore, a n-bit (cells) LFSR with period $2^n - 1$ can be determined with only $2n$ values.

B. Randomness Test

Randomness test is a frequently used data evaluation methodology, which is used to analyze the distribution pattern of a set of random data. For RFID tags, the test data are randomly generated PRNG binary sequences. These tests focus on a variety of different types of non-randomness that could exist in a sequence. There are many practical measures of randomness for a binary sequence. These include measures based on statistical tests, transforms, and complexity or a mixture of these. The focus of the test is the proportion of 0s and 1s for the entire sequence. NIST recommends to run the Frequency test first, since this supplies the most basic evidence for the existence of non-randomness in a sequence, specifically, non-uniformity [13]. If the Frequency test fails, the likelihood of other tests failing is high. The purpose of this test is to determine whether the number of ones and zeros in a sequence are approximately the same as would be expected for a truly random sequence. The test assesses the closeness of the fraction of 0s to 1s , that is, the number of 0s and 1s in a sequence should be almost the same. All subsequent are tested to check whether they can pass of statistical tests. Here, we provide some notions of the The Frequency Test which are used in our paper:

Item	Notation	Contents
Input	ε	The original input string of 0 and 1 bits
	n	The number of bits
Output	p-value	The probability
	S_n	The sum of the first n values of X_i

p-value is the probability (under the null hypothesis of randomness) that the chosen test statistic will assume values that are equal to or worse than the observed test statistic value when considering the null hypothesis. The p-value is frequently called the tail probability.

1) $\varepsilon = (\varepsilon_1, ..., \varepsilon_n)$

$$X_i = 2\varepsilon_i - 1 (1 \leq i \leq n) \quad (2)$$

$$S_n = X_1 + ... + X_n \quad (3)$$

2) Calcurate s_{obs}

$$s_{obs} = \frac{|S_n|}{\sqrt{n}} \quad (4)$$

3) S_n, Mean$\mu = 0$ variance $\sigma^2 = 0$

$$p - value = \mathrm{erfc}\Big(\frac{|\mathrm{S_n}|}{\sqrt{2\mathrm{n}}}\Big) \quad (5)$$

Furthermore, NIST test has Frequency Test within a Block, The Runs Test, Tests for the Longest-Run-of-Ones in a Block, the Binary Matrix Rank Test, the Non-overlapping template Matching Test, the Overlapping Template Matching Test, Maurer's Universal Statistical Test, the Linear Complexity Test, the Serial Test, the Approximate Entropy Test, the Cumulative Sums (Cusums) Test, the Random Excursions Test and the Random Excursions Variant Test.

III. Distinguish attack against J3Gen-like PRNG

Sage *et al.* [1] proposed a new PRNG (named J3Gen) for EPC Gen2 RFID Tags. J3Gen is based on a dynamic linear feedback shift register (DLFSR) of n cells. J3Gen consists of the following four main components.

- Truly random source: The technique used in J3Gen is the oscillator-based high frequency sampler by Che *et al.*
- Decoding Logic : It is the responsible for managing the internal PRNG clock of J3Gen.
- Polynomial Selector : It is the responsible for the linearity avoidance of J3Gen. A set of m primitive feedback polynomials are implemented as a wheel and it rotates one position if $trn = 0$ and two positions if $trn = 1$. These rotations are performed every l cycles $(1 \leq l < n)$.
- LFSR : The use of it makes an ideal system for both energy and computational constrained environments.

This first analysis shows that the security evaluation carried out by the authors 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}.

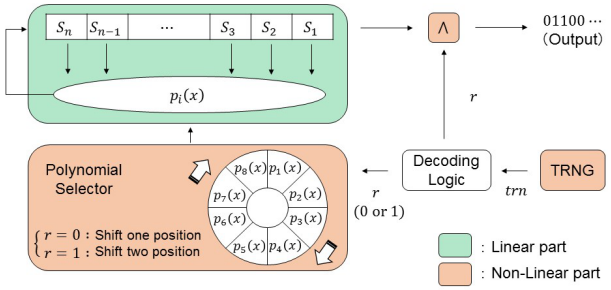

Fig. 1. Illustration of J3Gen

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.

Table 1 Feedback Polynomials ($m = 8$)

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

Joan *et al.* showed that J3Gen has the randomness criteria set by the EPCglobal Gen2 standard. However, J3Gen has two main disadvantages for generating randomness. The one is AND operation. This operation makes the outputs more bias. So, we tested with NIST SP800-22 to evaluated the statistic properties. The other is the vulnerability for failure of TRNG. We expect that this PRNG would not work if TRNG does not work properly, which is a common case for low cost RFID tags. For this reason, we modifies the J3Gen to achieve high quality as a PRNG for low cost RFID tags, so to explore J3Gen more in detail is necessary. In this section, we propose a distinguishing attack to show that how to distinguish the J3Gen output from a truly random output string.

In cryptography, a distinguishing attack is a useful and formal methods of evaluate the security of encryption schemes (especially for stream cipher and block cipher) or pseudo-random number generators, to show that it is not information-theoretically secure. An adversary can distinguish between the output of a particular cipher and the output of a truly random process with a non-negligible probability. In this section, we show that how an adversary can distinguish an random string from LFSR-based PRNG from a truly random string. Of course it is true that there is always a distinguishing attack against any algorithmic cipher; since it must have a finite key, and so brute-force key enumeration will yield a distinguishing attack of complexity 2^{k-1} where k is the key length.

The attack works as follows:

notations:
z_i: i-th keystream output bit
r_i: i-th true random input bit

Let's assume $r_1 = 0$, the r_1 will choose polynomial P_1 which is $P_1 = 1 + x + x^5 + x^6 + x^7 + x^{11} + x^{16}$. The taps correspond to $s_1, s_6, s_10, s_11, s_12$ and s_{16}. The keystream using these six states are as follows:

$$r_1 \wedge s_1 = z_1 \quad (6)$$
$$r_6 \wedge s_6 = z_6 \quad (7)$$
$$r_{10} \wedge s_{10} = z_{10} \quad (8)$$
$$r_{11} \wedge s_{11} = z_{11} \quad (9)$$
$$r_{12} \wedge s_{12} = z_{12} \quad (10)$$
$$r_{16} \wedge s_{16} = z_{16} \quad (11)$$

When p_1 is applied for the first time, s_{17} is generated as follows:

$$s_{17} = s_1 \oplus s_6 \oplus s_{10} \oplus s_{11} \oplus s_{12} \oplus s_{16}$$

and also we have

$$r_{17} \wedge s_{17} = z_{17} \quad (8)$$

$\wedge$ can be approximate by $\oplus$ with probability $(1/2 + 1/4)$ as a result, (1)-(6), (8) can be approximated as follows with probability $(1/2 + 1/4)$ each.

$$r_1 \oplus s_1 \oplus z_1 = 0 \quad (12)$$
$$r_6 \oplus s_6 \oplus z_6 = 0 \quad (13)$$
$$r_{10} \oplus s_{10} \oplus z_{10} = 0 \quad (14)$$
$$r_{11} \oplus s_{11} \oplus z_{11} = 0 \quad (15)$$
$$r_{12} \oplus s_{12} \oplus z_{12} = 0 \quad (16)$$
$$r_{16} \oplus s_{16} \oplus z_{16} = 0 \quad (17)$$

And also due the LFSR relationship, we have

$$r_{17} \oplus s_1 \oplus s_6 \oplus s_{10} \oplus s_{11} \oplus s_{12} \oplus s_{16} \oplus z_{17} = 0$$

Thus we can cancel the internal state s_i and derive $r_1 \oplus r_6 \oplus r_{10} \oplus r_{11} \oplus r_{12} \oplus r_{16} \oplus r_{17} \oplus z_1 \oplus z_6 \oplus z_{10} \oplus z_{11} \oplus z_{12} \oplus z_{12} \oplus z_{16} \oplus z_{17} = 0$ with probability $1/2 + 2^{(6-1)} * (1/4)^6 = (1/2 + 2^{-7})$

Now we have shown the strong bias of the event $r_1 \oplus r_6 \oplus r_{10} \oplus r_{11} \oplus r_{12} \oplus r_{16} \oplus r_{17}$
$\oplus z_1 \oplus z_6 \oplus z_{10} \oplus z_{11} \oplus z_{12} \oplus z_{12} \oplus z_{16} \oplus z_{17} = 0$. If the attacker can observe the true random number sequence r_i, then he needs around only $1/(2^{-7})^2 = 2^{14}$ outputs in order to distinguish it from the uniform distribution. Notice that we do not put any assumptions on the quality of the TRNG here, which means that the bias is generated due to the design of the PRNG.

In order to reveal the relation between the TRNG and PRNG, we can further consider the following two events

$$\Pr(r_1 \oplus r_6 \oplus r_{10} \oplus r_{11} \oplus r_{12} \oplus r_{16} \oplus r_{17}) = \frac{1}{2} + e_r \quad (18)$$

$$\Pr(z_1 \oplus z_6 \oplus z_{10} \oplus z_{11} \oplus z_{12} \oplus z_{16} \oplus z_{17}) = \frac{1}{2} + e_z \quad (19)$$

By applying the Piling-up Lemma again, we can derive

$$e_z = 2^{-8} \cdot e_r^{-1} \quad (20)$$

This provides a rough approximated relationship between the bias of TRNG and PRNG, which again explains that even the TRNG is perfect random, the PRNG outputs are definitely biased. Notice that (20) requires that (18) and (19) to be independent, which cannot be satisfied accurately in reality.

IV. Modified LFSR PRNGs

Our main design goal is to develop a more secure 16-bit LFSR-based pseudo-random number generator with as minimal cost as possible. Such lost cost designs are needed in many constrained environments, especially for EPC Gen2 RFID tags. The LFSR is initialized with some state, and the cipher has to stop running the moment the LFSR arrives to some predetermined state. We fixed the heavy biased output and sequent polynomial selector mothod proposed in the previous proposal by [1], [12]. 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.

A. Our Design Principle

For the general purpose of PRNG for EPC Gen2 RFID tags, we should design the mechanism which satisfies the following properties:

- Low Computational Cost: The computational overhead of generating the pseudo-random numbers be small due to the limited power available to RFID tags.
- Low Storage Requirement: The data stored in a RFID tag should be kept as small as possible since the tag memory is extremely constrained.

In this paper, we designed two proposals, proposal one and proposal two, in accordance with the above policies. We also provide two implementation options for trn, because the truly random source in low cost tags are usually unstable. Our schemes are simple and easy to be implemented in EPC Gen2 tags.

- Tag Type: Passive, without battery
- Design: Modified version of J3Gen (improved polynomial selection and bias of outputs)
- Resistance to failure: Our Proposals operate even if TRNG has failed.
- Implementation to Tag: Possible
- Components: TRNG, Decoding Logic, LFSR(s), Polynomial Selector(s), XOR
- Security: Satisfy the requirements of PRNG for RFID and Randomness testing (except for failure during)

B. Proposal One

Here, we introduce our first variant proposal of our modified multi-LFSR PRNGs. The basic idea of our proposal is to make the choosing from eight 16-bit LFSR more randomly. Different from the previous work, we choose not too use them rotationally and sequently. Our proposal one has the following characteristics. Multiple LFSRs can be defined, in turn, as an LFSR where the feedback polynomial, $p_i(x)$, is not static, but changes dynamically.

We modify the construction of J3Gen and propose a more secure PRGN. Our basic idea is to make the selection of the multiple polynomial more random and this can make the adversary more difficult to predict the linear behavior of the random bit generation. Future more we modified the output AND computation to a XOR computation, because the XOR function is perfectly balanced by observing an output value, there is exactly a 50% chance for any value of the input bits which is better J3Gen PRNG. This distinguishes the XOR gate from other Boolean functions such as the OR, AND or NAND gate. Moreover, AND and NAND gates are not invertible.

Regarding the physical source of randomness (trn), there are different proposals to derive true random sequences of bits from the hardware of a radio-frequency identification (RFID) tag. The technique that we use in our design is an oscillatorbased high frequency sampler by Che *et al.* [11], that offers high simplicity and suitability for EPC Gen2 designs. The output of the TRNG is fed to the Decoding Logic which, in turn, manages the Polynomial Selector.

In Sugei's J3Gen scheme, the feedback polynomials are implemented as a wheel, which rotates depending on the bit value given by the TRNG module. If the truly random bit is a logical 0, the wheel rotates one position, that is, it selects the next feedback polynomial. Instead, if the truly random bit is a logical 1, then the wheel rotates two positions, that is, the Polynomial Selector jumps one feedback polynomial and selects the next one.

In our proposal one, we modify as follows (Illustrated in Fig. 2):

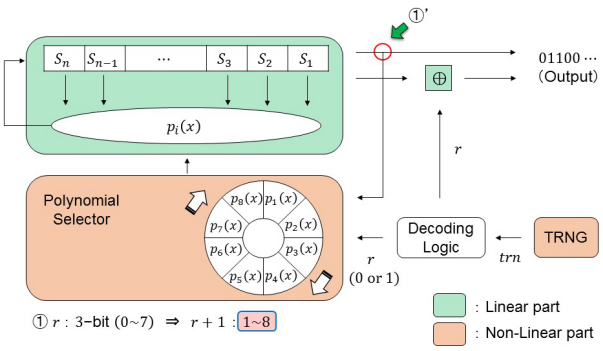

Fig. 2. Proposal One

In our proposal one, depending on the input seed bits, the output random bit r is either a zero ($s_i = 0$) or one ($s_i = 1$). If the seeds bit s_i behaves perfectly randomly, i.e., it is unpredictable and has exactly a 50% chance to have the value 0 or 1, then both possible random bit also occur with a 50% likelihood. By change AND logic to XOR logic, if we input the $trn = 1$, the output has less bias and independent from on the value of the seed bits, there is a 50% chance that the output random bit is either a 1 or a 0.

C. Proposal Two

Here, we introduce the second variant of our proposals, the Proposal two. The random bits are loaded into two part of registers (whose lengths depend on the block size). This proposal is inspired by a well-know light-weight stream cipher, KATANTAN [14]. Similar with KATANTAN, the random bits are generated during the LFSR processing rounds. For each round, several bits are taken from the registers and enter the mixing process or we can use two nonlinear Boolean functions, here we adapt the simple construct by adding internal randomness instead of employing the heavy non-linear functions process. The output of the Boolean functions is loaded to the least significant bits of the registers (after they were shifted).This should be done in an invertible manner. To ensure sufficient randomized mixing, the PRNG wait for several rounds of the output are executed.

In many low-end RFID applications, the seed of the internal PRNG is loaded once to tag and is never changed. In such instances, it should be possible to provide an good randomize solution which can handle a seed which is not stored in memory. We also have to consider the internal operation of XOR-Expression, a bitwise expression which only uses the XOR operator, the implementation is cheap for using XOR-epression. The degree of an XOR-Expression is the number of distinctly named variables in an expression. The expression $x \oplus y \oplus z$ has a degree of 3, the proposed PRNG aim at increasing the degree for the internal state randomization without increase the implementation cost. For some distinct bitwise variables inside the tags, we can get a reduced form if the form is expressed in the minimum degree which is still simple for security analysis.

Our proposal two is illustrated in Fig. 3. We can divide the LFSR into two part for low cost of implementation. However, to attain a stronger security condition, we should also can double the internal state of LFSRs and use two full 8 LFSRs. That will allow us to make full use of the randomness provided by 16 LFSRs.

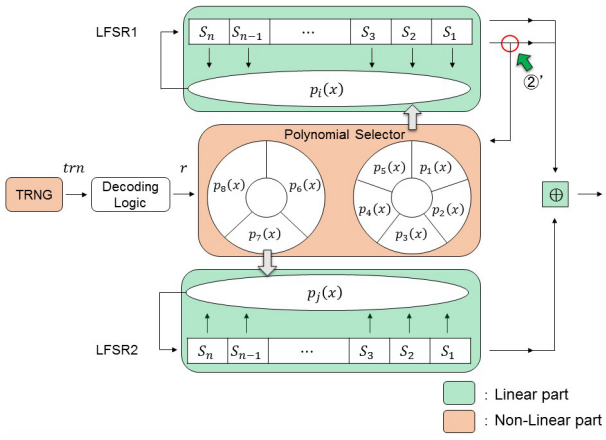

Fig. 3. Proposal Two

The reasons of our irreducible polynomial assignment is to achieved the efficient hardware implementation by choosing polynomials with several coefficients in common: two groups of polynomials share coefficients x_{16}, x_{11}, x_6, x_5 and x_0. This simplifies the logic circuitry (fewer gates). However, for security reason, we have to do the modification more 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. The size and design of each component of J3Gen implies a specific hardware implementation, being the LFSR size (n) and the number of implemented feedback polynomials on tag (m) the parameters that most significantly impact on the hardware complexity of our design, which is the same with J3Gen PRNG.

From the security point of view, the implementation should look for the different combinations of LFSRs within the hardware implementation boundaries to find the best security implementation for this purpose. The Polynomial Selector module includes the polynomials implementation and the logic hardware to select each polynomial. This hardware complexity is considerably low and suitable for our PRNG schemes.

V. Experimental Results and Security Analysis

In this section, we present our experimental results and compared our schemes with the existing scheme propose by [1], [2]. Our experiments perform randomness testing on our two proposal with NIST SP800-22 and our security analysis is based on the result of the experiment. In this experimental study, we conducted implementations following by the NIST SP800-22 test in the following environment. We simulate the RFID computation using desktop PC and we refer the LFSR implementation to the book written by Bruce Schneier [15]. Our experimental environment is as follows:

Item	Content
OS	Ubuntu 14.04 LTS (64 bit) Linux
Memory	8.00 GB
CPU	Intel Core i5-3320M 2.60 GHz
Language	C
Compiler	gcc

Here, in this paper, we apply the minimum Set [13] to test the pseudo-randomness on the LFSR-based PRNGs. Firstly, we want to make an observation on the proportion of zeroes and ones for the entire sequence which is generated from our proposed RRNG. The purpose of this test is to determine whether that number of ones and zeros in a sequence are approximately the same as would be expected for a truly random sequence. The test assesses the closeness of the fraction of ones to zero, that is, the number of ones and zeroes in a sequence should be about the same.

- Frequency Test within a Block: The focus of the test is the proportion of zeroes and ones within M-bit blocks. The purpose of this test is to determine whether the frequency of ones is an M-bit block is approximately $M/2$.
- Test for the Longest Run of Ones in a Block: To determine whether the length of the longest run of ones within the tested sequence is consistent with the length of the longest run of ones that would be expected in a random sequence. Note that an irregularity in the expected length of the longest run of ones implies that there is also an irregularity in the expected length of the longest run of zeroes. Long runs of zeroes were not evaluated separately due to a concern about statistical independence among the tests.
- Linear Complexity Test: To determine whether or not the sequence is complex enough to be considered random. Random sequences are characterized by a longer feedback register.
- Serial Test: Observe the frequency of each and every overlapping m-bit pattern across the entire sequence.
- Cumulative Sums (Cusum) Test: The focus of this test is the maximal excursion (from zero) of the random walk defined by the cumulative sum of adjusted (-1, +1) digits in the sequence. The purpose of the test is to determine whether the cumulative sum of the partial sequences occurring in the tested sequence is too large or too small relative to the expected behavior of that cumulative sum for random sequences. This cumulative sum may be considered as a random walk.

As follows, we show the results obtained after our NIST tests. The minimum pass rate for each statistical test, with the exception of the random excursion (variant) test approximately equals to 0.960150 for each sample size = 100 binary sequences. For each PRNG schemes, we show the results of the attack and summarize the comparison in Table 1. Here, we focus on two kinds of distinguish attack using input and output pair (α and β). α is the output from PRNG and to be used as input for the NIST randomness test, while β is the output obtained by XOR the i-th bit and $i + (1 - 8)$-th bits, we observe the bias. In other words, we evaluate the random when we obtain continuously more than 9 bits or more than 100 bits which is the recommended by NIST. The results obtained deviation of the output, we can see that our proposals have smaller bias.

Type	Dterministic attack	Distinguish attack α	Distinguish attack β	Test passed
LFSR	$2n$	$\frac{1}{2}$	$\frac{1}{2}$	within a block, cumulative sums
J3Gen	$2nm$	$\frac{3}{4}$	$\frac{3}{5}$	liner complexity
Proposal 1	$2^n - 1$	$\frac{1}{2}$	$\frac{1}{2}$	all
Proposal 2	$2^n - 1$	$\frac{1}{2}$	$\frac{1}{2}$	all

TABLE I. Experimental analysis

The other table showed in Fig. 4 is the comparison of vulnerability for failure of TRNG. In this table, we summarized the relation between correctness and distinguishable in our proposal when there is no trn input due to the tags' malfunction. We can see that it can be further improved by employing the non-linear processing.

From the results above, we can evaluate the proposals. It can produce arbitrarily long binary sequences and become a common tool for hardware and software based cryptographic random or pseudo-random number generators. These tests

	Type	Proportion of Output (Correct)	Distinguish Attack	Testing (Pass Test)
1	Proposal 1'	90 [%]	【Frequency of 0】 $\frac{1}{2}$ $(b = 0)$	・Within a Block ・Longest Run of Ones in a Block ・Linear Complexity ・Cumulative Sums
		70 [%]		
		50 [%]		・Within a Block ・Linear Complexity ・Cumulative Sums
		30 [%]	【Frequency of 0】 $\frac{501300}{10^6}$ $(b = 0.0026)$	・Within a Block ・Linear Complexity ・Cumulative Sums
		10 [%]	【Frequency of 0】 $\frac{501400}{10^6}$ $(b = 0.0028)$	
2	Proposal 2'	90 [%]	【Frequency of 0】 $\frac{1}{2}$ $(b = 0)$	・Within a Block ・Longest Run of Ones in a Block ・Linear Complexity ・Cumulative Sums
		70 [%]		
		50 [%]		
		30 [%]		
		10 [%]		

Fig. 4. The random test results of our proposals

focus on a variety of different types of non-randomness that could exist in a sequence. Our experimental analysis is as follows:

$$erfc(z) = \int_z^{\infty} \frac{2}{\sqrt{\pi}} \mathrm{e}^{-x^2} dx \tag{21}$$

- Mean : $\mu = nPr_{success} = 1000 \times (1 - \alpha) = 990$
- Variance : $\sigma^2 = nPr_{success}(1 - Pr_{success}) = 1000 \times (1 - \alpha) \times \alpha = 9.9$

$$0.99 \pm 3\sqrt{\frac{0.99 \times 0.01}{m}} \tag{22}$$

From the experimental analysis, we can see that our two proposals satisfy the criteria of the randomness required by EPC Gen2 tag application. We have also perform another test on the minimum criteria proposed by as follows.

1) Probability of a single 16-bit pseudo-random number R_{16}: Our experiments show that the probability that the 16-bit pseudo-random number generated by our two variant has value $R_{16} = j$, for any j, shall be bounded by

$$\frac{0.8}{2^{16}} < \Pr(j) < \frac{1.25}{2^{16}} \tag{23}$$

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 R_{16}s shall be less than 0.1%, regardless of when the tags are energized.
3) Probability of predicting an R_{16}: An R_{16} drawn from our tags' PRNGs are not predictable with a probability greater than 0.025% if the outcomes of prior draws from the RNG, performed under identical conditions, are known.

From the experimental analysis provided above, we can see that our proposals can achieve better PRNG security properties under the limitation of low-cost EPC Gen2 RFID tags.

A. Security analysis of proposals

In our proposed schemes, the state of the LFSR is uniquely determined by the 16 internal register bits. Before launching an attack, the adversary have to perform the polynomial detection in order to obtain the necessary information of which polynomial is being use and predict the PRNG output based on the polynomial. Our proposal can improve the security performance by thwarting the adversary to predict the current state of polynomial usage. In our proposals, the adversary in the synchronization step cannot predict the PRNG output since the adversary cannot have collected all m feedback polynomials due to the unpredictability of the trn value which is truly random. Hence, if trn is perfect, then the security is up to the best bound of the multiple polynomial.

Given a certain state, the LFSR deterministically assumes its next state. However, the deterministic characteristics can be randomized in our by internal random source of RFID tags, which means if fresh random bit is continually being added into the state of LSFR When initialized by a private internal state in LFSR, state, the tag starts to generate repeatedly. Since an 16-bit state vector can only assume 2^{15} nonzero states, the maximum sequence length before repetition is 2^{15}. Note that the all zero state must be excluded. If an LFSR assumes an all zero state, it will get stuck in it, i.e., it will never be able to leave it again. Our configurations of the irreducible polynomials $(p_1, ..., p_8)$ yield maximum length LFSRs. Our proposals resist forward tractability because our PRNGs use updated trn value in each output session. Therefore, attacking these random values does not help the attacker to use them since they are updated in each session. However, because all of the internal state approved and stored inside a tag, an attacker who wants to perform backward traceability needs to break the subsequent stages of authentication in order to corrupt into the tag. The tag's old values are replaced with new values in the updating phase, which helps the server to prevent any unknown values from gaining access. We will need more $2^{18} * e_r^2$ random bits outputs to distinguish from the random distribution.

For LFSR-based PRNGs, the polynomial generator from a simple 16-cell LFSR with period $2^{16} - 1$ can be determined with only $2n$ bits due to the linearity of this method, by using a system of n equations or the Berlekamp-Massey algorithm [16]. In [17], [18], the complexity of the attack against LFSR-based PRNG are investigated. This attack is polynomial when k is fixed. This attack will only be exponential in n, if $k = O(n)$. The number of bits used in a non-linear filter cannot be small. In practice, talking about polynomial (or not-polynomial) time is misleading and should always be confronted with concrete results. Knowing that the maximum degree of the filtering function cannot exceed k, the security bound of our design with linear feedback can be great than $\binom{n}{k}^{16}$, given $\binom{n}{k}$ bits of PRNG output, by simple linearization. For Proposal Two, the construction of PRNG can be more secure if we can combine several LFSRs and add a highly non-linear Boolean function f to destroy the linear relation inside the LFSR. Then the security of our proposals can be analysed in terms of correlation attacks, that can be seen as solving a system of multivariate linear equations, true with some probability.

VI. Conclusion

RFID is one of the key technologies that shows promising applications in the Internet of Things, especially in infrastructure security management, healthcare and retail business. In this paper, we studied a pseudo-random number generator (PRNG) design for EPC Gen2 RFID technologies called J3Gen and proposes two improved schemes. We proposed two pseudo-random number generators based on a linear feedback shift register (LFSR) configured with a multiple-polynomial tap architecture fed by a physical source of randomness, achieving a reduced computational complexity and low-power consumption as required by the EPC Gen2 standard. For lightweight PRNGs implemented in RFID tags, the main component is LFSR that generates sequences, it can be more secure if we can later transformed in a nonlinear fashion by a lightweight filter function. We applied the distinguish attacks on the current PRNGs with linear feedback for lightweight RFID tags and show that the adversary can distinguish output the existing schemes. We then provided two improved PRNGs to meet security requirement for lightweight RFID tags.

Acknowledgment

Part of this work was supported by JSPS KAKENHI 15K16004 and 15K16005.

References

[1] J. Melia Segui, J. Garcia Alfaro, and J. Herrera Joancomarti, "Multiple-polynomial lfsr based pseudorandom number generator for EPC Gen2 RFID tags," in *IECON 2011: 37th Annual Conference of the IEEE Industrial Electronics Society*, IEEE, Ed., 2011, pp. 3820 – 3825.

[2] J. Melia-Segui, J. Garcia-Alfaro, and J. Herrera-Joancomarti, "J3Gen: A PRNG for low-cost passive RFID," *Sensors*, vol. 13, no. 3, p. 3816, 2013. [Online]. Available: http://www.mdpi.com/1424-8220/13/3/3816

[3] G. E. Inc., "EPC radio-frequency identity protocols generation-2 UHF RFID," 2013. [Online]. Available: http://www.gs1.org/gsmp/kc/epcglobal/uhfc1g2

[4] S. Piramuthu, "Protocols for RFID tag/reader authentication," *Decision Support Systems*, vol. 43, no. 3, pp. 897 – 914, 2007, integrated Decision Support. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167923607000103

[5] A. Juels, "RFID security and privacy: A research survey," *IEEE J.Sel. A. Commun.*, vol. 24, no. 2, pp. 381–394, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1109/JSAC.2005.861395

[6] E. W. Lempel, A., "High speed generation of maximal length sequences," *IEEE Transactions on Computer*, vol. 2, no. 2, pp. 227–229, 1971.

[7] G. G. Kalikinkar Mandal, Xinxin Fan, "Warbler: A lightweight pseudorandom number generator for EPC C1 Gen2 tags," *Cryptology and Information Security Series, Radio Frequency Identification System Security.*, vol. 8, pp. 73–84, Sep. 2011.

[8] H. Martin, E. S. Millan, L. Entrena, P. P. Lopez, and J. C. H. Castro, "AKARI-X: A pseudorandom number generator for secure lightweight systems," *11th IEEE International On-Line Testing Symposium*, vol. 0, pp. 228–233, 2011.

[9] J. Wu and M. O'Neill, "Ultra-lightweight true random number generators," *Electronics Letters*, vol. 46, no. 14, pp. 988–990, July 2010.

[10] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, and A. Ribagorda, "LAMED - a prng for epc class-1 generation-2 RFID specification," *Comput. Stand. Interfaces*, vol. 31, no. 1, pp. 88–97, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1016/j.csi.2007.11.013

[11] W. Che, H. Deng, W. Tan, and J. Wang, "A random number generator for application in rfid tags," in *Networked RFID Systems and Lightweight Cryptography*, P. H. Cole and D. C. Ranasinghe, Eds. Springer Berlin Heidelberg, 2008, pp. 279–287. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-71641-9_16

[12] J. Melia Segui, J. Garcia Alfaro, and J. Herrera Joancomarti, "A practical implementation attack on weak pseudorandom number generator designs for EPC Gen2 tags," *Wireless personal communications*, vol. 59, no. 1, pp. 27 – 42, july 2011.

[13] L. E. Bassham, III, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, E. B. Barker, S. D. Leigh, M. Levenson, M. Vangel, D. L. Banks, N. A. Heckert, J. F. Dray, and S. Vo, "Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications," Gaithersburg, MD, United States, Tech. Rep., 2010.

[14] C. De Cannire, O. Dunkelman, and M. Kneevi, "KATAN and KTANTAN a family of small and efficient hardware-oriented block ciphers," in *Cryptographic Hardware and Embedded Systems - CHES 2009*, ser. Lecture Notes in Computer Science, C. Clavier and K. Gaj, Eds. Springer Berlin Heidelberg, 2009, vol. 5747, pp. 272–288. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04138-9_20

[15] B. Schneier, *Applied Cryptography (2Nd Ed.): Protocols, Algorithms, and Source Code in C*. New York, NY, USA: John Wiley & Sons, Inc., 1995.

[16] J. Massey, "Shift-register synthesis and BCH decoding," *IEEE Trans. Inf. Theor.*, vol. 15, no. 1, pp. 122–127, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1109/TIT.1969.1054260

[17] N. T. Courtois and W. Meier, "Algebraic attacks on stream ciphers with linear feedback," in *Proceedings of the 22Nd International Conference on Theory and Applications of Cryptographic Techniques*, ser. EUROCRYPT'03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 345–359. [Online]. Available: http://dl.acm.org/citation.cfm?id=1766171.1766200

[18] M. A. Orumiehchiha, J. Pieprzyk, and R. Steinfeld, "Cryptanalysis of WG-7 (a lightweight stream cipher for RFID encryption)," *IACR Cryptology ePrint Archive*, pp. 687–687, 2011.