Journal of Electronic Testing (2019) 35:519–527
https://doi.org/10.1007/s10836-019-05818-8

Security Analysis and Improvement of the Pseudo-random Number Generator Based on Piecewise Logistic Map

Dragan Lambić[1,2]

Received: 6 December 2018 / Accepted: 2 August 2019 / Published online: 9 August 2019

Abstract

In this paper, a security analysis of the pseudo-random number generator based on piecewise logistic map is made, which reveals the existence of a serious problem. Although the assumed safety of this pseudo-random number generator (PRNG) is estimated at 2^{136}, it is possible to carry out successful brute-force attack whose complexity is about 2^{103}. Furthermore, the attack on the analyzed PRNG based on a known sequence of output bits is presented which can reduce the complexity of the brute-attack to about 2^{95}. The examples of both attacks are provided in this paper. For the above mentioned reasons, the analyzed PRNG cannot be considered safe for the use in cryptographic systems. An improved version of the analyzed PRNG is proposed, which can eliminate the perceived shortcomings.

Keywords Chaos · Pseudo-random number generator · Cryptanalysis · Cryptography · Image encryption · Video encryption

1 Introduction

Chaos has been widely used for secure communications and encryption which enabled the development of a number of chaotic cryptosystems [10]. Chaotic systems are often used as the core component of PRNG [4, 22]. PRNGs have a wide application, especially in cryptographic systems developed for encryption of image and video files which require a large amount of randomly generated bits. Cryptographic properties of a PRNG are very important because cryptographic security of stream encryption schemes depends largely on quality of a produced pseudo-random sequence [22].

In recent years a great number of chaos based PRNGs have been proposed. However, a significant percent of these PRNGs have serious shortcomings which makes them inadequate for the use in secure cryptographic systems. The key space of chaos-based PRNGs could be very vulnerable in the case when the secret key is used for generation of control parameters of the underlying chaotic system. For example, in the pseudo-random bit generator based on multi-modal maps, initial values of chaotic map are used as a secret key but some features of the used multi-modal map caused the existence of a certain number of weak keys [11]. The inadequate selection of interval for control parameters of quantum chaotic map in PRNG [2] caused a situation in which 99 percent of the key space is composed of weak keys [9]. In the efficient chaos PRNG applied to video encryption, initial conditions of two chaotic maps were used as a part of a secret key, but a considerable number of these conditions leads to the fixed points which reduced the security of this PRNG [10]. Moreover, it is demonstrated that a stream cipher based on the symbolic dynamics of the logistic map and tent map can satisfy certain security conditions only in the case when the parameters of these maps are selected conveniently [4]. Otherwise, in the case of a random selection of parameters, this stream cipher can suffer from some cryptographic weaknesses [4].

In many cases, the features of chaotic map used in cryptographic system are the main cause of its security deficiencies [3]. For example, the logistic map was used extensively in PRNG design but a significant number of such PRNGs have some security problem [12–14, 17, 18].

The logistic map has one parameter $\mu \in [0, 4]$ whose value significantly affects the behaviour of this chaotic

Responsible Editor: S. Bhunia

✉ Dragan Lambić
dragan.lambic@tdtu.edu.vn

[1] Department for Management of Science and Technology Development, Ton Duc Thang University, Ho Chi Minh City, Vietnam

[2] Faculty of Mathematics & Statistics, Ton Duc Thang University, Ho Chi Minh City, Vietnam

map. When the value of μ is closer to 4, there is a greater probability that the non-chaotic behaviour of this continuous-space chaotic map will be avoided. For this reason, it is desirable to use $\mu = 4$ or values which are very close to 4 but in this way the key space of the logistic map for this parameter is significantly reduced which diminishes the applicability of this chaotic map in cryptography. The logistic map has two fixed points regardless of value of its parameter, which can also cause some security problems in certain cryptographic applications.

Apart from the aforementioned chaotic properties there are other features of the logistic map which indicate its inadequacy for cryptographic applications [5]. Efficiency of the encryption process depends greatly on the uniformity of a probability of occurrence of values from each subinterval of the domain in which applied chaotic map is defined. Values near fixed points of the logistic map are obtained more often than other values which negatively affects efficiency of cryptographic systems based on Logistic map [5].

Another significant problem of the logistic map is the relation between the maximal length of orbits and value of the parameter μ. Depending on the type of cryptographic system there are various ways in which an attacker can estimate the maximal length of orbits of a chaotic map used. When an attacker knows the maximum value of orbit length, he or she can estimate value of the control parameter μ and therefore reduce the security of a cryptographic system [5].

The return map is used to show the relationship between two consecutive values of some chaotic map. The return map can provide valuable information regarding some characteristics of chaotic map. The return map of the logistic map provides sufficient information required for the calculation of value of the control parameter μ. For example, in the paper [21] it is demonstrated how chosen cipher-text attack can be used in order to obtain the return map which provided sufficient information for estimation of the value of the control parameter μ.

Another important characteristic of the logistic map which could be used by an attacker is structural complexity. The statistical complexity of the logistic map is closely related to the value of the control parameter μ [5]. The statistical complexity of the logistic map is almost a bijective application of the control parameter and it decreases when the control parameter μ increases [5].

All previously mentioned features of the logistic map indicate that the control parameter of the logistic map μ should not be used as a secret parameter of some cryptographic system. Therefore, the only parameter of this continuous-space chaotic map which can be used as a part of the secret key of some cryptographic system is its initial value. For this reason new versions of the logistic map with variable parameter μ are developed.

In some cases inadequate design of PRNG or cryptographic system is the main cause for low security, regardles of the features of used chaotic map. For example, initial condition and control parameter of the pseudorandom number generator based on the pseudorandomly enhanced logistic map, are calculated on the basis of a secret key. However, the inadequate design enabled the relations between different values of the secret key which caused that approximately 2^{50} different secret keys produce the same initial condition and control parameter of the enhanced logistic map, which makes this PRNG non-resistant to a brute-force attack [8]. The inadequate design of the image cipher based on the chaotic standard and logistic maps leads to the creation of equivalent cipher which enabled the chosen plain-text attack [18]. Although the mentioned cipher was modified in order to resist the chosen plain-text attack, further research has shown that the modified version is still insecure [12, 13]. The above mentioned examples indicate that key space of the existing PRNGs and their design should be thoroughly analyzed from the aspect of security, in order to eliminate possible shortcomings which could enable cryptographic attacks.

In [22] a pseudorandom number generator based on piecewise logistic map is proposed. The proposed PRNG is designed for the application in a stream cipher for secure communication, but the security analysis revealed the existence of a serious security problem. The detected security problems of PRNG [22], which are described in this paper, are not directly caused by the piecewise logistic map but they are a consequence of the auxiliary mechanism which is used to change value of the parameter μ. If the attacker has found a weakness in the cryptosystem which can be exploited with the complexity lower than an estimated key space, then we can consider that the analyzed cipher is broken [20]. PRNG must have more than 2^{128} different secret keys in order to resist a brute-force attack [6]. Although the safety of PRNG presented in [22] is estimated at approximately 2^{136}, it is possible to carry out a successful brute-force attack with the complexity of less than 2^{128}.

The main contribution of this paper is the security analysis of the pseudo-random number generator based on piecewise logistic map [22] which has shown that estimated security of this PRNG, which was not cryptanalized before, is much higher than its real security level. In order to demonstrate the deficiencies of this PRNG, a brute-force attack and an attack based on known sequence of output bits are presented. Presented attacks are important for two reasons. First, users of PRNG [22] will know the real security level of this PRNG and therefore can decide whether this security is appropriate for their intended applications. Second, designers of PRNGs will know that this particular auxiliary mechanism (and similar ones) used to change

value of some secret parameter of a chaotic map is not secure enough and therefore should be avoided in this form in PRNG design.

A secondary contribution of this paper is the proposed improvement of the analyzed PRNG which could serve as a quick patch to prevent proposed attacks until users of this PRNG switch to a new PRNG. Bearing in mind that this improvement is designed only with the goal to prevent attacks presented in this paper, its application in systems which require very high security is not recommended before more thorough tests prove that it is completely safe.

The rest of this paper is organized as follows. In Section 2, the analyzed PRNG is described. The security analysis and examples of the attacks on the analyzed PRNG are presented in Section 3. In Section 4, the improvement of the analyzed PRNG is presented. The performance analysis of the proposed improvement is presented in Section 5. The conclusions are drawn in Section 6.

2 Description of the Analyzed PRNG

The analyzed PRNG is based on the piecewise logistic map (PLM) which is proposed and described in the paper [22]. PLM is defined by

$$x_{i+1} = PLM(x_i) =$$

$$= \begin{cases} N^2\mu x_i(\frac{1}{N} - x_i), & \text{for } 0 < x_i < \frac{1}{N} \\ 1 - N^2\mu(x_i - \frac{1}{N})(\frac{2}{N} - x_i), & \text{for } \frac{1}{N} < x_i < \frac{2}{N} \\ \vdots & \\ N^2\mu(x_i - \frac{j-1}{N})(\frac{j}{N} - x_i), & \frac{j-1}{N} < x_i < \frac{j}{N} \\ 1 - N^2\mu(x_i - \frac{j}{N})(\frac{j+1}{N} - x_i), & \frac{j}{N} < x_i < \frac{j+1}{N} \\ N^2\mu(x_i - \frac{N-2}{N})(\frac{N-1}{N} - x_i), & \frac{N-2}{N} < x_i < \frac{N-1}{N} \\ 1 - N^2\mu(x_i - \frac{N-1}{N})(1 - x_i), & \frac{N-1}{N} < x_i < 1 \\ x_i + \frac{1}{100N}, & x_i = 0, \frac{1}{N}, ..., \frac{N-1}{N} \\ x_i - \frac{1}{100N}, & x_i = 1 \end{cases} \quad (1)$$

State value of the PLM is $x_i \in (0, 1)$, the control parameter is $\mu \in (0, 4]$ and N is the number of segments of the PLM. Secret key of the analyzed PRNG consists of the initial status value $x_0 \in (0, 1)$ and parameters $\mu_0 \in (0.01, 3.99)$, $m \in (0.01, 0.1)$ and $R_0 \in [0, 255]$ [22]. The key space of this PRNG is estimated at $9.17 \cdot 10^{40} \approx 2^{136}$, when the precision of a floating-point number is 10^{-13} [22]. For the precision of a floating-point number of 10^{-15}, the estimated key space is approximately 2^{156}. PRNG proposed in [22] is described by following steps.

1. Choose the PLM with $N = 64$. Set the initial values of the secret key x_0, μ_0, m, R_0.
2. Iterate the PLM one time. Transform the current status value x_i of the PLM to the binary format. Use 8 bits, 9th to 16th bits after the decimal point, to obtain integer K_i.
3. Calculate $P_i = K_i \oplus S(R_i)$, where $S(R_i)$ returns the substitution value of R_i according to the AES S-box [1].
4. Update the value of R by $R_{i+1} = K_i$.
5. Adjust the value of μ based on the following rules:
   (i) $\mu_{i+1} = \mu_i + \frac{P_i}{256}$
   (ii) if $\mu_{i+1} > (4 - m)$ then $\mu_{i+1} = \mu_{i+1} - (4 - m)$
   (iii) if $\mu_{i+1} < m$ then $\mu_{i+1} = m$
6. Omit the first 1024 generated numbers in order to get rid of the transient effect of the chaotic map. If the sufficient quantity of pseudorandom numbers have been generated, stop this PRNG. Otherwise, go to Step 2 to generate another 8-bit pseudorandom number P_i.

3 Security Analysis

The security level of the analyzed PRNG will be demonstrated based on its resistance to a brute-force attack and an attack based on known sequence of output bits.

3.1 The Brute-force Attack on the Analyzed PRNG

It is very important that the secret key is selected in a random manner. If there is any connection between the parameters, which are parts of the secret key, the attacker can take advantage of this connection in order to speed up the process of cryptanalysis. In step 5 of the analyzed PRNG [22], a very strong connection between parameters μ_i and m is established, which significantly accelerates the brute-force attack.

In step 5, if $\mu_{i+1} < m$ then the value of parameter μ_{i+1} is set to m ($\mu_{i+1} = m$) in order to restore the parameter μ_{i+1} in the set of allowed values $\mu \in (m, 4 - m]$. For that reason the attacker can guess the secret key x_{i+1}, m, R_{i+1} instead of x_0, m, μ_0, R_0 (because some $\mu_{i+1} = m$) which reduces the complexity of the brute-force attack for 2^{45} times which represents the complexity of the search for μ_i when the precision of a floating-point number is 10^{-13} (or 2^{51} times for precision of 10^{-15}). Here we must bear in mind that the attacker must repeat the attack for each $i > 0$ until $\mu_{i+1} = m$ is found, which to some extent slows the described attack.

In order to determine how many attempts are approximately required to find the case in which $\mu_i = m$, testing of the analyzed PRNG was conducted. The number of the required attempts largely depends on the value of the parameter m. When the value of m is lower, the probability of finding the case in which $\mu_i = m$ is also lower. For this reason,

tests were conducted for $m = 0.01$ and $m = 0.1$. For both values of parameter m, 200 tests with different values of P_i were conducted. For $m = 0.01$ the maximal value of i for which $\mu_i = m$ was found is $i = 2599$ and the average value of i is 530. For $m = 0.1$ the maximal value of i for which $\mu_i = m$ was found is $i = 223$ and the average value of i is 38.

In order to estimate the total complexity of the described attack, the worst case scenario for the attacker will be used. The maximal number of attempts to find the case in which $\mu_i = m$ was $2599 < 2^{12}$. Therefore the total complexity of the brute-force attack on the secret key x_i, m, R_i is approximately $\frac{2^{136} \cdot 2^{12}}{2^{45}} = 2^{103}$ when the precision of a floating-point number is 10^{-13}. In the case when the precision of a floating-point number is 10^{-15}, the total complexity of the brute-force attack on the secret key x_i, m, R_i is approximately $\frac{2^{156} \cdot 2^{12}}{2^{51}} = 2^{117}$. In both cases, the estimated complexity of the attack is less than 2^{128} recommended in [6].

3.2 The Example of the Brute-force Attack on the Analyzed PRNG

In this section example of the brute-force attack on analyzed PRNG [22] will be described. Assume that the user of the analyzed PRNG randomly selected the initial values $x_0 = 0.519067023060468$, $\mu_0 = 2.881509362057897$, $R_0 = 201$ and $m = 0.09$ which will be considered as a secret key. Based on these initial values, the user generates a sequence of pseudo random numbers by using the analyzed PRNG. During this process, the user must calculate the values of μ_i for $i \geq 0$ which are shown in Table 1.

The attacker does not know any of the secret values, but know that some value μ_i will eventually be equal to m. For this reason, the attacker first tries to find the secret key $x_0, m, \mu_0 = m, R_0$ by checking all the combinations of the values x_0, m, R_0 in order to generate the pseudorandom number P_0. Because $\mu_0 \neq m$, the attacker cannot generate P_0 and therefore must continue the search. The attacker repeats this procedure for all $x_i, m, \mu_i = m, R_i$ until the pseudorandom number P_i is successfully generated. In Table 1 it can be seen that $\mu_{18} = m = 0.09$. Therefore, after checking all the combinations of the values x_{18}, m, R_{18} the attacker can successfully generate the pseudorandom number P_{18} and therefore find out the values of the secret key $x_{18} = 0.211869763952096, m = 0.09, \mu_{18} = 0.09, R_{18} = 149$. This secret key enables the attacker to calculate all subsequent values of pseudorandom numbers. The complexity of this search when the precision of a floating-point number is 10^{-13} was $19 \cdot \frac{2^{136}}{2^{45}} < 2^{96}$. In the case when the precision of a floating-point number is 10^{-15}, the total complexity of the brute-force attack on the secret key x_{18}, m, R_{18} is approximately $\frac{19 \cdot 2^{156}}{2^{51}} < 2^{110}$. In both cases, the estimated complexity of the attack is less than 2^{128} recommended in [6].

Table 1 Calculation of μ_i

i	μ_i	P_i	$\mu_i + \frac{P_i}{256}$
0	2.8815093620579	10	2.9205718620579
1	2.9205718620579	81	3.2369781120579
2	3.2369781120579	107	3.6549468620579
3	3.6549468620579	209	4.4713531120579
4	0.561353112057897	76	0.858228112057897
5	0.858228112057897	51	1.0574468620579
6	1.0574468620579	157	1.6707281120579
7	1.6707281120579	182	2.3816656120579
8	2.3816656120579	127	2.8777593620579
9	2.8777593620579	10	2.9168218620579
10	2.9168218620579	253	3.9051031120579
11	3.9051031120579	54	4.1160406120579
12	0.206040612057897	166	0.854478112057897
13	0.854478112057897	113	1.2958843620579
14	1.2958843620579	216	2.1396343620579
15	2.1396343620579	200	2.9208843620579
16	2.9208843620579	161	3.5497906120579
17	3.5497906120579	96	3.9247906120579
18	0.09	33	0.21890625
19	0.21890625	109	0.6446875

3.3 The Attack on the Analyzed PRNG Based on Known Sequence of Output Bits

An important feature of cryptographically secure PRNG is that an attacker cannot obtain information on a secret key based on known sequence of output bits. Unfortunately, based on sequence of 8 output bits P_i of the analyzed PRNG, the attacker can calculate value of K_i which represents 8 bits, 9th to 16th bits after the decimal point of part of the secret key x_i.

Assume that the attacker knows the values of some 8 output bits P_i of the analyzed PRNG. Because there is no information about cryptosystem in the paper [22], which is intended to use the bits generated by the analyzed PRNG, we can only assume the way in which the attacker can find out the values of the sequence of output bits. It is fully expected that the attacker can obtain the sequence of output bits of the analyzed PRNG based on a certain number of known plaintext bits.

The attacker guesses the value of part of the secret key R_i and based on step 3 of the analyzed PRNG [22] calculates the value of K_i by using the formula $K_i = P_i \oplus S(R_i)$, where $S(R_i)$ returns the substitution value of R_i according

to the AES S-box [1]. Now the attacker knows the values of 8 bits of x_i (9th to 16th bits after decimal point) and needs to guess only the remaining bits of x_i (in addition to μ_i and m). For this reason, the complexity of the search for the secret key x_i, m, μ_i, R_i is $\frac{2^{136}}{2^8} = 2^{128}$ when the precision of a floating-point number is 10^{-13} or $\frac{2^{156}}{2^8} = 2^{148}$ when the precision of a floating-point number is 10^{-15}. When used alone, this attack does not lead to a more serious threat to the analyzed PRNG security, because the required complexity of the attack is still equal to or higher than 2^{128} guesses. However, this attack can further reduce the complexity of the attack described in the previous subsection.

Assume that the attacker knows the values of first $8 \cdot (i + 1)$ output bits of the analyzed PRNG $P_0, ..., P_i$. The attacker guesses the value of R_i and calculates the value of K_i which reduces the complexity of the search for x_i by 2^8 times. Then the attacker guesses the remaining bits of x_i and m, assuming that $\mu_i = m$ which further reduces the complexity of the search for 2^{45} or 2^{51} times depending on the precision of a floating-point number. The maximal number of attempts to find the case in which $\mu_i = m$ was estimated at $2599 < 2^{12}$. Therefore the total complexity of the brute-force attack on the secret key x_i, m, R_i is approximately $\frac{2^{136} \cdot 2^{12}}{2^{45} \cdot 2^8} = 2^{95}$ when the precision of a floating-point number is 10^{-13}. In the case when the precision of a floating-point number is 10^{-15}, the total complexity of the brute-force attack on the secret key x_i, m, R_i is approximately $\frac{2^{156} \cdot 2^{12}}{2^{51} \cdot 2^8} = 2^{109}$. In both cases, the estimated complexity of the attack is less than 2^{128} recommended in [6]. For this reason, the analyzed PRNG [22] cannot be considered safe for the use in cryptographic systems.

3.4 The Example of the Attack on the Analyzed PRNG Based on Known Sequence of Output Bits

In this section, an example of the attack on analyzed PRNG [22] based on known sequence of output bits will be described. In this example, the same initial values as in the previous example will be considered as the secret key $x_0 = 0.519067023060468$, $\mu_0 = 2.881509362057897$, $R_0 = 201$ and $m = 0.09$. Based on these initial values, the user generate sequence of pseudo random numbers P_i presented in Table 1, by using the analyzed PRNG.

Assume that the attacker knows the values of first $8 \cdot 20 = 160$ output bits of the analyzed PRNG $P_0, ..., P_{19}$. Also, the attacker knows that some value μ_i will eventually be equal to m. First, the attacker guesses the value of R_0 and based on the known value of $P_0 = 10$ calculates the value of K_0 which represents 9th to 16th bits after the decimal point of x_0. After that the attacker tries to guess the remaining bits of x_0 and the value of $\mu_0 = m$. However, because $\mu_0 \neq m$, the attacker cannot recover the secret key and therefore must continue the search for $i > 0$. The attacker repeats this procedure for all P_i until whole secret key is found.

In Table 1 it can be seen that $\mu_{18} = m = 0.09$. The attacker knows the value of $P_{18} = 33$ and checks all possible values of R_{18}. When $R_{18} = 149$, the attacker calculates bits of K_{18} by using the formula $K_{18} = P_{18} \oplus S(R_{18})$ and obtains a binary representation of $K_{18} = 11101000$. Because K_{18} represents 9th to 16th bits after the decimal point of x_{18} the attacker only needs to guess the remaining bits of x_{18} and the value of $\mu_{18} = m$. In this way, the attacker recovers values of the secret key $x_{18} = 0.211869763952096, m = 0.09, \mu_{18} = 0.09, R_{18} = 149$ which enables the calculation of all the subsequent values of pseudorandom numbers P_i for $i > 18$.

The complexity of this search when the precision of a floating-point number is 10^{-13} was $19 \cdot \frac{2^{136}}{2^{45} \cdot 2^8} < 2^{88}$. In the case when the precision of a floating-point number is 10^{-15}, the total complexity of the brute-force attack on the secret key x_{18}, m, R_{18} is approximately $19 \cdot \frac{2^{156}}{2^{51} \cdot 2^8} < 2^{102}$. In both cases, the estimated complexity of the attack is less than 2^{128} recommended in [6].

4 The Improvement of the Analyzed PRNG

Although the security of the analyzed PRNG is significantly threatened, the perceived shortcomings can be removed. The improvement of the PRNG proposed in [22] is described by the following steps.

1. Choose the PLM with $N = 64$. Set the initial values of the secret key x_0, μ_0, m, R_0.
2. In order to generate one value of K_j, two iterations of PLM x_{2j}, x_{2j+1} will be used in the following way. Iterate the PLM one time. Transform the current status value x_{2j} of the PLM to the binary format. Calculate the value of $v = floor(x_{2j} \cdot 32)$ where $floor(y)$ denote a function which map a real number y to the largest integer less than or equal to y. Let b denote the bits of x_{2j} after the decimal point. Use 4 bits $b_{v+8}b_{v+9}b_{v+10}b_{v+11}$ of x_{2j} as first four bits of integer K_j. Iterate the PLM once again and transform the value x_{2j+1} to the binary format. Calculate the value of $w = floor(x_{2j+1} \cdot 32)$ and use 4 bits $b_{w+8}b_{w+9}b_{w+10}b_{w+11}$ of x_{2j+1} as second four bits of integer K_j.
3. Calculate $P_j = K_j \oplus S(R_j)$, where $S(R_j)$ returns the substitution value of R_j according to the AES S-box [1].
4. Update the value of R by $R_{j+1} = K_j$.
5. Adjust the value of μ based on the following rules:
   (i) $\mu_{j+1} = \mu_j + \frac{P_j}{256}$

(ii) if $\mu_{j+1} > (4-m)$ then $\mu_{j+1} = \mu_{j+1} - (4-m)$
(iii) if $\mu_{j+1} < m$ then $\mu_{j+1} = \mu_{j+1} + m$

6. Omit the first 1024 generated numbers in order to get rid of the transient effect of the chaotic map. If the sufficient quantity of pseudorandom numbers has been generated, stop this PRNG. Otherwise, go to Step 2 to generate another 8-bit pseudorandom number P_j.

In the proposed improvement, only the necessary changes are made in order to avoid the perceived shortcomings, while at the same time most of the features of the original PRNG are preserved. For this reason, steps 1, 3, 4 and 6 have remained unchanged. In step 2, two important changes were made. First, two iterations of PLM are used in order to obtain integer K_j, instead of one. In this way, based on step 3, the attacker can discover only four bits of some x_i instead of eight bits in the original version of the analyzed PRNG. Although the attacker can also discover four bits of the next iteration of PLM x_{i+1}, these bits cannot be used in order to recover the value of x_i. Second, the positions of bits from x_i used to obtain integer K_j are not fixed as in the original version of the analyzed PRNG, but they depend on value of x_i. The attacker can calculate the values of four bits of x_i, based on step 3, which could reduce the complexity of the attack on x_i by 2^4 if the positions of these bits were fixed. However, because there are $32 = 2^5$ different positions in which four bits from K_j could be located in x_i, the attacker cannot use this information to reduce the search complexity.

In step 5, the rule (iii) from the original version of the analyzed PRNG is changed to:

(iii) if $\mu_{j+1} < m$ then $\mu_{j+1} = \mu_{j+1} + m$.

In this way, a significant relationship between parameters μ_j and m is avoided. Therefore, the attacker is forced to guess the whole key x_{2j}, m, μ_j, R_j, because the situation in which $\mu_{j+1} = m$ is highly unlikely. Based on the above changes the perceived shortcomings of the original version of the analyzed PRNG are removed. Therefore, the security of this improvement of the analyzed PRNG is estimated at 2^{136} when the precision of a floating-point number is 10^{-13} and 2^{156} in the case when the precision of a floating-point number is 10^{-15}. This security level satisfies the general requirement for resisting brute-force attack [6].

Also, the security of the improved PRNG can be additionally improved by using additional parameters. Step 5 could be completely changed to $\mu_{j+1} = f(\mu_j)$ where f is some chaotic map. For example, any of the chaotic maps [7, 15, 16] can be used with certain modifications. In this way, the parameters of used chaotic map could be considered as additional part of a secret key, so the complexity of the brute-force attack would be increased.

5 Performance Analysis of the Proposed Improvement

In this section a randomness analysis of the proposed improvement will be conducted in order to determine whether the proposed changes to the original PRNG affected the randomness of the generator. The randomness of the original version of the analyzed PRNG was determined according to the NIST 800-22 test suite so the same test will be applied to the improved version. Also, some important characteristics of the original and improved version of the analyzed PRNG will be compared.

5.1 Randomness Analysis

The NIST test suite is a standard library of tests which is used for statistical testing of randomness of binary sequences [19]. The NIST test suite consists of 15 tests which are aimed to detect various regularities in bit sequences. In several papers, including the paper [22] in which the analyzed PRNG is proposed, the randomness of some PRNG is evaluated by using NIST library in the following way. A certain number of sequences are generated and tested, and afterwards the fraction of sequences that pass each test are calculated. In order to confirm the randomness of a generator the fraction should be from the interval

$$(1-\alpha) \pm 3\sqrt{\frac{\alpha}{n}} \tag{2}$$

where $\alpha = 0.01$ and n is the number of tested sequences. In order to determine the randomness of the improved version of the analyzed PRNG, 1,000 sequences of 1,000,000 bits each are tested by using the NIST test suite. The results are shown in Table 2.

Random excursions, random excursions variant, and non-overlapping template tests are actually families of subtests which consist of more than one test. Random excursions and random excursions variant consist of 8 and 18 tests respectively (which is a reasonable amount of data) so all results could be included in Table 2. The non-overlapping template consists of 148 tests which is a significant amount of data so only the minimum and maximum proportions are presented in Table 2. All proportions for the non-Overlapping template test are in the interval [0.984, 0.997].

Also, random excursions and random excursions variant tests have certain limitations which prevented testing of the whole sample. These tests are not applicable to the binary sequences with an insufficient number of cycles which is below 500. For this reason only 631 samples with the number of cycles exceeding 500 were evaluated for

Table 2 Results of the NIST statistical tests

Test name	Number of sequences with P value ≥ 0.01 (success)	Average P value	Proportion of successful sequences	Result
Approximate entropy	988	0.185048	0.988	Success
Frequency within a block	997	0.447995	0.997	Success
Cumulative sums forward	987	0.508602	0.987	Success
Cumulative sums reverse	985	0.381264	0.985	Success
Discrete Fourier transform	988	0.331273	0.988	Success
Frequency	990	0.450657	0.990	Success
Longest run in a block	993	0.613824	0.993	Success
Non-overlapping template min	984	0.194456	0.984	Success
Non-overlapping template max	997	0.835112	0.997	Success
Overlapping template	993	0.287577	0.993	Success
Binary matrix rank	993	0.744921	0.993	Success
Runs	997	0.661632	0.997	Success
Serial 1	989	0.488778	0.989	Success
Serial 2	987	0.318319	0.987	Success
Linear complexity	995	0.517693	0.995	Success
Maurer's universal	992	0.301948	0.992	Success
Random excursions (sample size 631)				
x=-4	626	0.650356	0.992	Success
x=-3	623	0.388204	0.987	Success
x=-2	624	0.797707	0.989	Success
x=-1	622	0.501301	0.986	Success
x=1	629	0.790598	0.997	Success
x=2	622	0.382221	0.986	Success
x=3	629	0.738245	0.997	Success
x=4	625	0.267224	0.990	Success
Random excursions variant (sample size 631)				
x=-9	624	0.470731	0.989	Success
x=-8	622	0.550339	0.986	Success
x=-7	628	0.719477	0.995	Success
x=-6	624	0.616291	0.989	Success
x=-5	625	0.423868	0.990	Success
x=-4	625	0.629319	0.990	Success
x=-3	628	0.751553	0.995	Success
x=-2	628	0.830396	0.995	Success
x=-1	623	0.939853	0.987	Success
x=1	625	0.461538	0.990	Success
x=2	625	0.423049	0.990	Success
x=3	623	0.665131	0.987	Success
x=4	623	0.676577	0.987	Success
x=5	622	0.741656	0.986	Success
x=6	623	0.821879	0.987	Success
x=7	626	0.856535	0.992	Success
x=8	626	0.519009	0.992	Success
x=9	623	0.601558	0.987	Success

these tests [22]. According to the results from Table 2, we can claim that the proposed improvement of the analyzed PRNG successfully passed all NIST tests due to fact that all proportions are within the interval defined by the Eq. 2. Therefore, the proposed improvement of the original PRNG can be used for pseudo-random number generation.

5.2 Comparison of the Analyzed and the Improved PRNG

In this section, the analyzed PRNG [22] and its improvement proposed in this paper will be compared according to some basic criteria for which there is available data in paper [22]. The comparison results are summarized in Table 3.

The ability to produce a pseudo-random sequence is an essential characteristic of every PRNG. According to the NIST test suite, both versions of the analyzed PRNG (original and improved) satisfy this requirement. Another very important property of a random number generator is its speed which depends on a number of operations required to obtain one bit or certain number of bits which represent the output of one iteration of a generator. The original and improved version of the analyzed PRNG produce 8 bits per one iteration. The original version of the analyzed PRNG require only 18 basic operations in order to generate 8 bits. The improved version of the analyzed PRNG require 32 operations in order to generate 8 bits, which is almost double compared to the original. However, this cost in the complexity and speed is small if we consider that the improved PRNG is much safer than the original PRNG. Bearing in mind that the original version of the analyzed PRNG has a very good reported speed of about 164 Mbit/s [22], the speed of the improved version (which is almost two time slower) should be sufficient for every application of the original PRNG.

Table 3 Comparison of the analyzed and the improved PRNG

	Analyzed PRNG [22]	Improved PRNG
NIST	Pass	Pass
ine Basic operations per iteration	18	32
ine Security level precision 10^{-13}	2^{95}	2^{136}
ine Security level precision 10^{-15}	2^{109}	2^{156}
ine Resistance to the attack based on known sequence of output bits	No	Yes
ine Resistance to the brute-force attack	No	Yes

When we consider security levels of the original and improved versions of the analyzed PRNG, the data from Table 3 show that the improved version of the analyzed PRNG has much better security than the original version. Regardless of the precision used to represent floating point numbers, the advantage of the improved version with respect to security is so great that it fully justifyies the reduced speed. When the precision of 10^{-13} is used, the improved version is 2^{41} times safer than the original version of the analyzed PRNG. When the precision of 10^{-15} is used, the security advantage is even greater because the improved version is 2^{47} times safer than the original version of the analyzed PRNG.

If we consider safety against attacks based on known sequence of output bits, the original version of the analyzed PRNG suffers from the reduced security by 2^8 times if the attacker knows sequence of 8 output bits. On the other hand, the improved version of the analyzed PRNG is immune to this type of attack and its security is the same regardless of the attacker's knowledge of output bits.

When the safety of a PRNG against a brute-force attack is in question, a PRNG must have at least 2^{128} different secret keys in order to resist this type of attack [6]. Although the key space of the original version of the analyzed PRNG [22] has more than 2^{128} different secret keys regardless of a precision used, due to the connection between parameters μ_i and m established in step 5, the number of possible keys is reduced below the acceptable level after a certain number of iterations. Because all the values of $\mu_{i+1} < m$ are set to a single value m it means that all the keys x_0, m, μ_0, R_0 after $i + 1$ iterations generate sequences which are equivalent to sequences generated by a much shorter key x_{i+1}, m, R_{i+1}. Therefore, the original version of the analyzed attack cannot resist a brute force attack on the secret key x_{i+1}, m, R_{i+1}. On the other hand, the improved version of the analyzed PRNG has the same size of key space, which is larger than 2^{128}, regardless of the number of iterations. Therefore we can consider that the improved version of the analyzed PRNG is safe against a brute-force attack.

6 Conclusion

In this paper, a security analysis of the pseudorandom number generator based on piecewise logistic map [22] is presented. The security analysis revealed some defects in the design of this PRNG which cause serious problems. Although the assumed safety of this pseudo-random number generator (PRNG) is estimated at 2^{136}, it is possible to carry out a successful brute-force attack with the complexity

of about 2^{103}. If an attacker knows some values of P_i, this attack can be further reduced to the complexity of about 2^{95}. For this reason, the analyzed PRNG cannot be considered safe for the use in cryptographic systems. In order to eliminate the perceived shortcomings, the improved version of the analyzed PRNG is proposed which satisfies the general security requirements.

Compliance with Ethical Standards

Conflict of interests The author declares that he has no conflict of interest.

References

1. Announcing the advanced encryption standard (AES). Federal Information Processing Standards Publication. 197.2001.2
2. Akhshani A, Akhavan A, Mobaraki A, Lim S-C, Hassan Z (2014) Pseudo random number generator based on quantum chaotic map. Commun Nonlinear Sci Numer Simulat 19:101–111
3. Alvarez G, Amigo JM, Arroyo D, Li S (2011) Lessons learnt from the cryptanalysis of chaos-based ciphers Lj Kocarev, Lian S (eds)
4. Arroyo D, Alvarez G, Amigo JM, Li S (2011) Cryptanalysis of a family of self-synchronizing chaotic stream ciphers. Commun Nonlinear Sci Numer Simul 16:805–813
5. Arroyo D, Amigo JM, Li S, Alvarez G (2010) On the inadequacy of unimodal maps for cryptographic applications. In: Ferrer JD, Balleste AM, Roca JC, Gomez AS (eds) XI Reunion Espanola sobre Criptologa y Seguridad de la Informacion (XI RECSI), Universitat Rovira i Virgili, Tarragona, Spain, pp 37–42, ISBN 978–84–693–3304–4
6. Ecrypt II yearly report on algorithms and keysizes; 2010. http://www.ecrypt.eu.org/documents/D.SPA.13.pdf
7. Lambić D (2015) A new discrete chaotic map based on the composition of permutations. Chaos, Solitons & Fractals 78:245–248
8. Lambić D (2017) Cryptanalyzing a novel pseudorandom number generator based on pseudorandomly enhanced logistic map. Nonlinear Dyn 89:2255–2257
9. Lambić D (2018) Security analysis and improvement of the pseudo-random number generator based on quantum chaotic map. Nonlinear Dyn 94:1117–1126
10. Lambić D (2018) Security analysis of the efficient chaos pseudo-random number generator applied to video encryption. J Electron Test 34:709–715
11. Lambić D (2018) Security analysis of the pseudo-random bit generator based on multi-modal maps. Nonlinear Dyn 91:505–513
12. Li C, Li S, Lo KT (2011) Breaking a modified substitution–diffusion image cipher based on chaotic standard and logistic maps. Commun Nonlinear Sci Numer Simul 16:837–843
13. Li CQ, Xie T, Liu Q, Cheng G (2014) Cryptanalyzing image encryption using chaotic logistic map. Nonlinear Dyn 78(2):1545–1551
14. Liu Y, Fan H, Xie EY, Cheng G, Li C (2015) Deciphering an image cipher based on mixed transformed logistic maps. Int J Bifurcation Chaos 25(13):1550188
15. Lorenz EN (1963) Deterministic non-periodic flow. J Atmos Sci 20(2):130–141
16. May RM (1976) Simple mathematical models with very complicated dynamics. Nature 261:459–465
17. Persohn K, Povinelli R (2012) Analyzing logistic map pseudorandom number generators for periodicity induced by finite precision floating-point representation. Chaos Solitons & Fractals 45(3):238–245
18. Rhouma R, Solak E, Belghith S (2010) Cryptanalysis of a new substitution-diffusion based image cipher. Commun Nonlinear Sci Numer Simul 15(7):1887–1892
19. Rukhin A, Soto J, Nechvatal J, Smid M, Barker E, Leigh S, Levenson M, Vangel M, Banks D, Heckert A, Dray J, Vo S (2001) A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST special publication pp 800–22
20. Schneier B (1996) Applied Cryptography. Wiley, New York
21. Skrobek A (2008) Approximation of a chaotic orbit as a cryptanalytical method on Baptista's cipher. Phys Lett A 372(6):849–859
22. Wang Y, Liu Z, Ma J, He H (2016) A pseudorandom number generator based on piecewise logistic map. Nonlinear Dyn 83:2373–2391

Dragan Lambić received the PhD degree from the Faculty of Mathematics in Belgrade. His primary research areas are mathematics, computer science, cryptography, chaos, computer science education and mathematics education.