Improvement of CPRNG of the PM-DC-LM Mode and Comparison with its Previous Version

Petr Zacek, Roman Jasek, David Malanik
Faculty of Applied Informatics
Tomas Bata University
Zlin, Czech Republic
e-mail: {zacek, jasek, dmalanik}@fai.utb.cz

***Abstract*—This paper presents the last results from our research focused on proposing the polymorphous mode of operation of block ciphers. The first attempt was based on Chaotic Pseudo-Random Number Generator (CPRNG) using logistic maps and it is called Polymorphous Mode − Deterministic Chaos – Logistic Maps mode (PM-DC-LM). CPRNG controls the polymorphous behavior of this mode. The CPRNG returns two values, *g* and d. Value g represents the last three digits of the number generated by CPRNG and the value *d* represents the last digit. Based on these two values, function F is controlled. In the initially proposed actual version, the CPRNG was limited in generating values ending by digits from one to nine. Thus, this leads to a non-optimal probabilistic distribution of values *g* and *d*. Therefore, an improvement is necessary. This paper shows the principle for improving the CPRNG in the PM-DC-LM mode and how to ensure the whole interval of values for g and d is generated, including numbers ending with zero. The principle is applied on the CPRNG and then it is tested. The differences between the actual and the upgraded version of the PM-DC-LM are described using the probabilistic distribution of generated values *g* and *d*. The entropies for values *g* and *d* of actual and upgraded versions are also calculated. These calculations are based on one million samples of values *g* and *d*.**

Keywords - Deterministic Chaos; Logistic Map; CPRNG; Symmetric Cryptography; Block Cipher; Block Cipher Mode of Operation; PM-DC-LM.

I. INTRODUCTION

Our designed Polymorphous Mode (PM) with its subversion PM-DC-LM was described by P. Zacek et al. [6] even though it was not named this way in that publication. This paper concentrates on testing one part of PM-DC-LM, namely CPRNG.

The motivation of our paper is to test the behavior of the used CPRNG built on a deterministic chaos – logistic maps and discuss the possibilities of the CPRNG used.

We discuss the possibilities of how to derive the initialization values for CPRNG from the initialization vector (*IV*). The last part shows the testing of CPRNG on real data. Real data were two random generated *IV* of 256-bit length. The values x_n were calculated for the first one million values x_n. The comparison of the appropriateness of CPRNG is based on entropies calculated from the probabilistic distributions of values *d* and *g*, which are generated by CPRNG.

Section 2 provides a brief description of the PM-DC-LM mode. Section 3 describes the internal logic of the used CPRNG. Section 4 is about the properties and possibilities of the CPRNG. In Section 5, the testing of the CPRNG is provided. The paper ends with a conclusion, in Section 7.

II. DESCRIPTION OF PM-DC-LM MODE

The acronyms "PM", "DC", and "LM" mean it is "Polymorphous Mode" with using the "Deterministic Chaos", and "Logistic Maps" as the type of deterministic chaos. This name depends on the type of the deterministic chaos and PM is used as the basis. The mode is mostly described in [6], where it was designed for the first time. Because of the polymorphous behavior of this mode, we changed the equation of the calculation for the next key. In contrast to the original, we used one more value named *g*. Value *g* is derived as the last three digits of value x_n from CPRGN.

III. DESCRIPTION OF THE USED CPRNG

The used CPRNG is described in this section. The CPRNG is fully described, and it is the same as in the material. It is based on the deterministic chaos logistic map, which operates in the following equation

$$x_n = r x_{n-1}(1 - x_{n-1}) \tag{1}$$

where *r* is the control parameter, x_n is the actual value and x_{n-1} is the previously generated value. If value *r* is a real number above 3.57, the system behaves chaotically for most of the values. Based on this fact, we choose real numbers in the interval <3.9, 4.0) to avoid numbers near the values of 3.82842712... In this generator, value *r* is generated from *IV*. Value x_i should be a real number from the interval (0, 1). The first value $x \rightarrow x_0$ is calculated using the following equations, where the first one is for values of $r > 3.9$ and the second is for $r = 3.9$ [2][6].

$$x_0 = 10(r - 3.9) \tag{2}$$

$$x_0 = 10^{-15} \tag{3}$$

The value r is calculated from the initialization vector before the encryption. The algorithm how to calculate the

 ISBN: 978-1-61208-493-0

value from the initialization vector is also described in [6][7].

IV. PROPERTIES OF PROPOSED CPRNG AND POSSIBILITIES

Because of the polymorphous structure of the proposed mode, the parts could be changed. According to CPRNG, the following features can be changed:

- We can change the type of deterministic chaos.
- We can change the principle how to calculate the values of r and x_0 as their precision.

A. Type of deterministic chaos

As the other parts could be changed, the type of the deterministic chaos could also be changed. Consequently, the way how to derive the initial values for CPRNG must be changed, too.

B. Derivation of values r and x_0

The actual derivation of value r is as follows:, (also described in the material [6])

1. Express IV as a binary number.
2. Split the IV into the same blocks of the length as precision p.
3. Represent it as numbers 0 and 1.
4. Calculate the sum of numbers 0 and 1 at their corresponding position modulo 10.
5. Concatenate sums as digits after to 3.9.

Ideally, we want to have different CPRNG for all IVs. The above principle is not the best. It leads to collisions. For example, if we have two IVs IV_1 = 0110 and IV_2 =1001, the value r is the same -> r = 3.91111. There are two ways to avoid the mentioned collisions.

1. Using a non-collision hash function before converting the IV into value r.
2. Representing the IV as a decimal number with fixed length (including leading zeros) and then concatenating all digits after value r.

These two possibilities have advantages and disadvantages as well. The first possibility should be secure if the non-collision function is used. It should also be quicker, and problems with the representation of long float numbers will happen less frequently. The second possibility does not lead to collisions at all, but we need to operate with long float numbers. For example, the value r computed from IV 256-bit length has 79 digits after the decimal point, so the representation of these numbers could be a problem [6][7].

The derivation of value x_0 could be changed as well.

- We can independently derivate it to value r.
  - Using another different algorithm for derivation of the value r from the same IV
  - Using the second IV for value x_0

V. TESTING OF CPRNG

A. Values d and g

We tested the first ten values from two different generators of deterministic chaos based on two randomly generated different IVs with length 256 bits, which were different in the last bit.. The precision p (number of digits in the value r after 3.9) of the r was chosen as 14 (maximal in the Python 3.x, which was used for testing). The results are shown below.

IV_1 = 111111…1000
IV_2 = 111111…1001

r_1 from IV_1 = 3.911132271809247
r_2 from IV_2 = 3.911132271809248

x_1 = 0.11132271809247
x_2 = 0.11132271809248

The first ten values generated by generator number one based on values r_1 and x_1:
[0.3869282003850097, 0.9277783349901382,
0.26206814046285243, 0.7563677304170022,
0.720726194443333, 0.787232496718318,
0.6551048496504891, 0.8836909470612287,
0.40199109175011855, 0.9402137244001353]

The first ten values generated by generator number one based on values r_2 and x_2:
[0.38692820038504017, 0.9277783349901654,
0.2620681404627615, 0.756367730416833,
0.7207261944436724, 0.7872324967177322,
0.6551048496518054, 0.8836909470596319,
0.40199109175491127, 0.94021372440381]

The first ten values g returned by generator number one and the corresponding first ten values d (digits highlighted in red color).
[097, 382, 243, 022, 333, 318, 891, 287, 855, 353]
The first ten values g returned by generator number two and the corresponding first ten values d (digits highlighted in red color).
[017, 654, 615, 833, 724, 322, 054, 319, 127, 381]

As can be seen from the results above, the first ten values g from generator number two are 100% different from the first ten values g from generator number one. The value d is different in 90% of the cases. So, although the IVs

 ISBN: 978-1-61208-493-0

are different by only one bit, the generators behave quite differently.

The ideal probabilistic distribution of values *d* and *g* should be indistinguishable from the probabilistic distribution of the random values. For the value of *d*, the probability should be 1/10 for any number from 0 to 9 which is unreachable, because using the deterministic chaos is always distinguishable from the random distribution. For demonstration, the CPRNG was tested on the first million values of *x*. The results are represented by the graphs below.

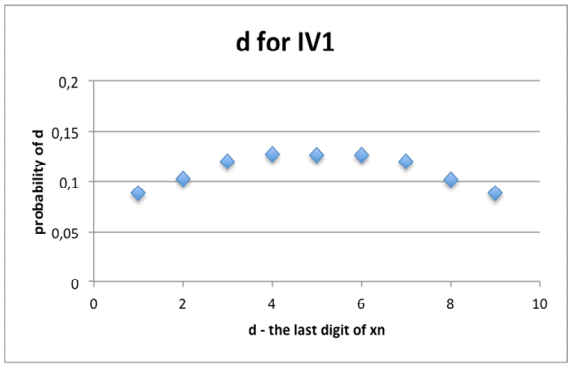

Figure 1. Probabilities for *d* of *IV1* [7]

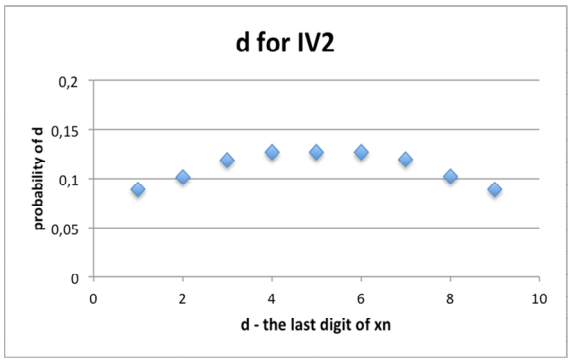

Figure 2. Probabilities for *d* of *IV2* [7]

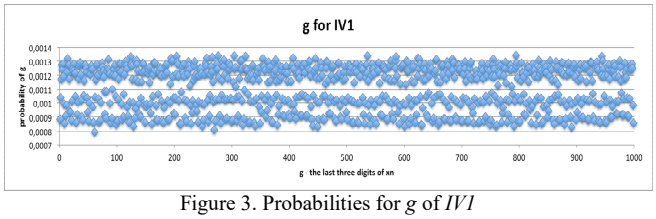

Figure 3. Probabilities for *g* of *IV1*

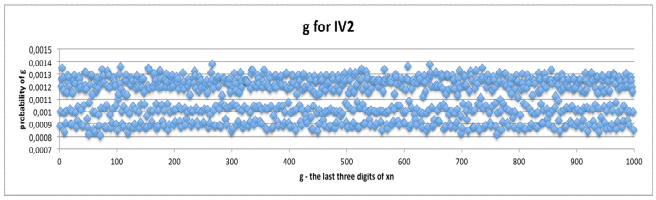

Figure 4. Probabilities for *g* of *IV2*

Figures 1 and 2 show the deterministic appearance of the probabilistic distribution for the values *d* generated by CPRNGs while the probabilities look more or less similar for both *IVs*. Figures 3 and 4 show the probabilistic distribution for the values *g* generated by CPRNGs. Probabilities oscillate around the most suitable value 0.001, concretely from 0.0008 to 0.0014. For more random-looking distribution, another type of pseudo-random number generator should be used or the CPRNG should be improved.

The probabilistic distribution shown in Figures 3 and 4 could be rewritten in more detail in the following Table 1 using values c.

TABLE I. PART OF PROBABILISTIC DISTRIBUTION OF *G* AS OCCURRENCE *C* OF *G* IN *IV1* AND *IV2* (*G* FROM 951 TO 970)

g	*c* for *IV1*	*c* for *IV2*	*g*	*c* for *IV1*	*c* for *IV2*
951	864	857	961	898	894
952	1008	1002	962	977	1006
953	1199	1177	963	1222	1248
954	1257	1250	964	1230	1307
955	1238	1239	965	1285	1293
956	1274	1294	966	1265	1287
957	1176	1162	967	1190	1216
958	1023	1054	968	1010	1072
959	900	861	969	923	855
960	0	0	970	0	0

Non-randomness of probabilistic distribution can be seen in detail in Table 1, where probabilities for values *g* from 951 to 970 are represented by numbers *c* according to the occurrences of *g* per one million generated values by CPRNG. The number *c* should be approximately 1000 in the perfect distribution. As can be seen, *g* values ending by digits different from zero are the only possible outcomes of CPRNG. Thus, the value of *g*; the last three digits from generated number x = 0.26206814046285240 will be 524 instead of 240. Based on this fact, potential occurrences for values *g* ending by zero are distributed among the others. Consequently, based on this fact and deterministic behavior of CPRNG it leads to maximums in the distribution. For proper and stable distribution, the CPRNG should be changed accordingly to generate the entire range of *g* values including numbers ending in zero.

B. Entropy for values g and d

The degree of randomness of CPRNG could be measured by evaluating the entropy for values *g* and *d* based on their probabilistic distribution. The entropy for values *g* was calculated fro (4) and for values *d* using (5).

$$H(g) = -\sum_{g=000}^{999} p_g . \log\left(p_g\right) \qquad (4)$$

$$H(d) = -\sum_{d=0}^{9} p_d . \log(p_d) \quad (5)$$

The entropy should be maximal. The maximum for the entropy for values *g* could be calculated from (6) and for values *d* from (7).

$$H(g)_{\max} = -(0.001 \cdot \log(0.001) \cdot 1000) \cong 6.9077 \quad (6)$$

$$H(d)_{\max} = -(0.1 \cdot \log(0.1) \cdot 10) \cong 2.3026 \quad (7)$$

On the other hand, the minimal entropy is 0. When (4) is applied to the probabilistic distribution for values *g,* obtained from the first one million values x from CPRNG based on the *IV1,* the entropy *H* is around 6.793. The entropy for values *g* based on the *IV2* is similar, 6.792.

Similarly, the entropy *H* for the probabilistic distribution for values *d*, obtained from CPRNG based on the *IV1,* is 2.1878 and for the *IV2* the entropy is 2.1877.

Based on these results, using the CPRNG may be assigned as sufficient and useful for generating pseudo-random numbers.

VI. POSSIBILITIES AND IMPROVEMENTS

There are many possibilities how to improve CPRNG for the PM-DC-LM or generally for the PM. The first possibility is the counting with values *g* and *d* ending in zero. This could be achieved by representing values *g* as the second, the third, and the fourth digit from the end of values *x* and similarly the second digit from the end as values *d*. According to it, the function F in the PM-DC-LM should be changed to count with the different values *d*. This means, the function F should be able to operate with 10 states instead of 9.

A. Evaluation of improvement – probabilistic distribution

In this section, the utilization of the proposed possible improvement is evaluated and shown. The testing of updated CPRNG was done on the same two randomly generated values *IV1* and *IV2* as before. The probabilistic distributions for values *d* obtained from the improved CPRNG are shown in Figures 5 and 6.

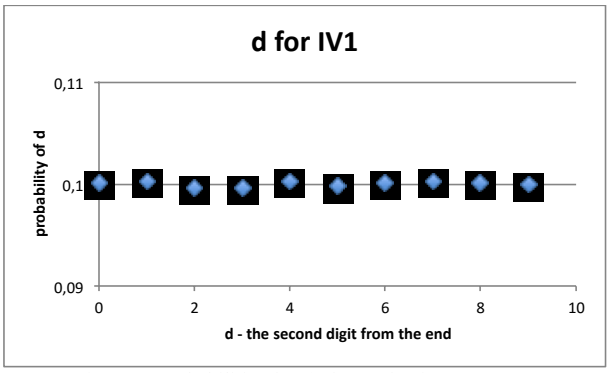

Figure 5. Probabilities for *d* of *IV1* after improvement

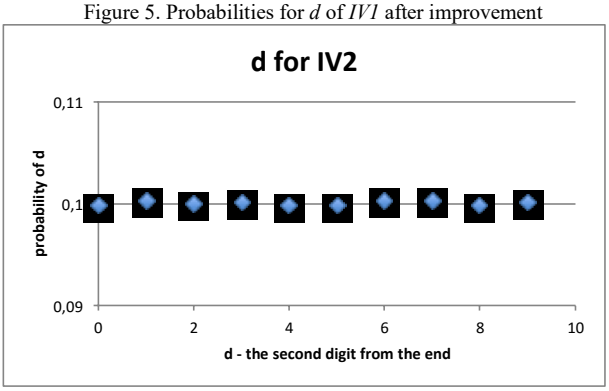

Figure 6. Probabilities for d of IV1 after improvement

As can be seen from Figures 5 and 6, the probabilistic distributions after improvement are closer to the random distribution. The detailed view is shown in Table 2 below as values *c*, where value c stands for a number of the occurrence of digit *d* in the first one million generated values *x*.

TABLE II. OCCURRENCE OF *D* IN *X*

d	*c* for *IV1*	*c* for *IV2*
0	100055	99801
1	100215	100243
2	99688	99908
3	99582	100087
4	100308	99769
5	99731	99842
6	100168	100239
7	100255	100290
8	100049	99738
9	99949	100083

As we can see from Table 2, the occurrence for values c is closer to the ideal. The probabilistic distributions of values *g* are shown in Figures 7 and 8.

 ISBN: 978-1-61208-493-0

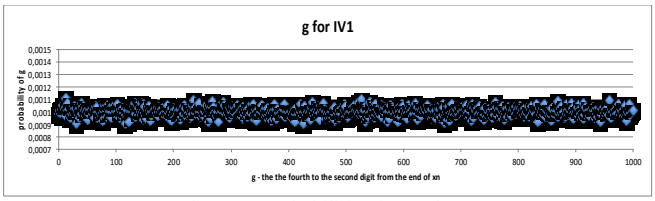

Figure 7. Probabilities for *g* of *IV1*

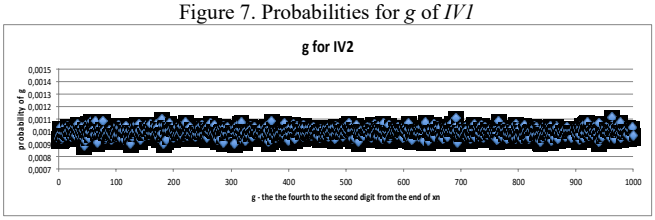

Figure 8. Probabilities for *g* of *IV2*

As we can see from Figures 5 and 6, the probabilistic distributions are getting closer to the optimal probability 0.001. The oscillation around the ideal value is compressed and optimized.

The probabilistic distribution shown in Figures 3 and 4 could be rewritten more detailed using the values *c*.

TABLE III. PART OF PROBABILISTIC DISTRIBUTION OF *G* AS OCCURRENCE *C* OF *G* IN *IV1* AND *IV2* (*G* FROM 951 TO 970)

g	*c* for *IV1*	*c* for *IV2*	*g*	*c* for *IV1*	*c* for *IV2*
951	1026	1027	961	952	1006
952	1029	1034	962	988	1113
953	967	995	963	1025	1024
954	980	1025	964	996	1050
955	1040	1018	965	1014	980
956	988	989	966	1004	1038
957	979	985	967	977	945
958	1091	1017	968	991	992
959	988	950	969	993	993
960	955	1002	970	1023	1017

As we can see in Table 3, after improvement, all values *g* could be obtained from CPRNG. Thus, the limitation in values ending by zero was removed. In the next section, the calculation of the entropies will be done as the confirmation of the improvement.

B. *Evaluation of improvement – entropy*

The maximal entropies are the same as (6) and (7) show. When (4) is applied to the new values of the probabilistic distribution for values *g* of the improved CPRNG, the entropies are $H = 6.9072$ for *IV1* and $H = 6.9073$ for *IV2*. Similarly, with using (5), the entropies for values *d* are $H = 2.3026$ for *IV1* and $H = 2.3026$ for *IV2*.

C. *Comparison after and before improvement*

The CPRNG is compared based on the entropies for values d and g after and before the improvement. The random (pseudo-random) generators should have the entropy close to its maximal value. The entropies for designed CPRNG used in the PM-DC-LM are summarized in Table 4.

TABLE 4. COMPARISON OF ENTROPIES OF CPRNG AFTER AND BEFORE IMPROVEMENT

	H(d)max	H(d)	H(d)improved	H(g)max	H(g)	H(g)improved
for IV1	2.30258509	2.18778925	2.30258215	6.90775527	6.79255874	6.90724147
for IV2	2.30258509	2.18774009	2.30258304	6.90775527	6.79248131	6.90725673

As we can see from Table 4, the entropies of improved CPRNG are closer to their maximal value for both probabilistic distributions for values *g* and *d*.

VII. CONCLUSION

In this paper, the CPRNG based on logistic maps used in our mode PM-DC-LM was tested. Some possibilities and ways for upgrading and changing the CPRNG were discussed. One of the possible mentioned ways, namely how to set CPRNG on two random generated *IV*s, was tested. The CPRNG was run one million times, and the probabilistic distribution of values *g* and *d* were shown in graphs. Consequently, the entropies for probabilistic distribution were calculated and compared with the maximal value of their corresponding entropies.

The CPRNG is distinguishable from the random distribution, as long as it is deterministic, whereas whole distribution looks sufficient and balanced even so before the improvement. After improvement, the entropies were calculated, and all were compared in Table 4. When the CPRNG was improved, a more random distribution was achieved than before.

For future research, it would be sensible to try to change the type of deterministic chaos or try to change the CPRNG to another general used PRNG. It would also be interesting to change the algorithm determining how the values *r* and x_n are derived from *IV*. We are planning to do further security analysis of the used CPRNG algorithm using the NIST statistical test suite and NIST randomness tests.

ACKNOWLEDGMENT

This work was supported by the Tomas Bata University Internal Grant Agency, Project No.: IGA/FAI/2015/47; further, it was supported by financial support from the Ministry of Education of the Czech Republic research project NPU I No.: MSMT-7778/2014; as well as by the European Regional Development Fund, CEBIA-Tech Project No.: CZ.1.05/2.1.00/03.0089

REFERENCES

[1] M. S. Bellovin. Columbia University. Modes of Operation Columbia, USA, 2009 [cit. 30.1.2015]. Online at: https://www.cs.columbia.edu/~smb/classes/s09/l05.pdf [accessed June 2016]

[2] E. W. Weisstein. "Logistic Map." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/LogisticMap.html [accessed June 2016]

[3] Modes Development. In: National Institute of Standards and Technology: Computer Security Resource Center [online]. 2001, http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html [accessed June 2016]

 ISBN: 978-1-61208-493-0

[4] J. C. Sprott, Chaos and Time-Series Analysis, Oxford University Press, 2003

[5] R. Senkerik, M. Pluhacek, I. Zelinka, D. Davendra, and Z. Oplatkova, “A brief survey on the chaotic systems as the pseudo random number generators”,, in Interdisciplinary Symposium on Complex Systems, vol 14. Emergence, Complexity and Computation (ISCS 2014),, Springer International Publishing, 2015, pp. 205-214, doi:10.1007/978-3-319-10759-2_22

[6] P. Zacek, R. Jasek, and D. Malanik, “Using the Deterministic Chaos in Variable Mode of Operation of Block Ciphers”, in Artificial Intelligence Perspectives and Applications, R. Silhavy, et al., Editors. 2015, Springer International Publishing. p. 347-354, DOI: 10.1007/978-3-319-18476-0_34

[7] P. Zacek, R. Jasek, and D. Malanik, “Possibilities and Testing of CPRNG in Block Cipher Mode of Operation PM-DC-LM”, ICNAAM 2015 - 13th International Conference of Numerical Analysis and Applied Mathematics, in press.

 ISBN: 978-1-61208-493-0