Constructing empirical tests of randomness

Marek Sýs, Petr Švenda, Martin Ukrop, Vashek Matyáš
Masaryk University, Botanicka 68a, Brno, Czech Republic
syso, svenda, matyas@fi.muni.cz, mukrop@mail.muni.cz

Keywords: eSTREAM; genetic programming; random distinguisher; randomness statistical testing; software circuit

Abstract: In this paper we introduce a general framework for automatic construction of empirical tests of randomness. Our new framework generalises and improves a previous approach (Švenda et al., 2013) and it also provides a clear statistical interpretation of its results. This new approach was tested on selected stream ciphers from the eSTREAM competition. Results show that our approach can lay foundations to randomness testing and it is comparable to the Statistical Test Suite developed by NIST. Additionally, the proposed approach is able to perform randomness analysis even when presented with sequences shorter by several orders of magnitude than required by the NIST suite. Although the Dieharder battery still provides a slightly better randomness analysis, our framework is able to detect non-randomness for stream ciphers with limited number of rounds (Hermes, Fubuki) where both above-mentioned batteries fail.

1 INTRODUCTION

We usually test randomness using batteries of empiricall test of randomness. Problem of the batteries is that they implement a fixed set of tests and can detect only limited types of patterns. Batteries detect only evident trivial defects and there is no problem to find sequences with some type of pattern (other than the tested ones) that pass all tests in the test battery. Since cryptographic functions have a deterministic output, it is a priori clear that they cannot pass all possible tests of randomness and there must exist empirical tests of randomness that reveal sequences as non-random.

In this work we introduce a novel framework for constructing empirical tests of randomness that can succeed in finding such a test (at least hypothetically). Our goal is to find an empirical test of randomness that indicates a given sequence is either non-random (with a high probability) or sufficiently random. In our framework, we iteratively construct randomness tests that adapt to the processed sequence. The construction is stochastic and uses genetic programming. Tests are constructed from a predefined pool of operations (building blocks). Set of operations, together with a limit for the number of operations, allows us to control the complexity of tests. The framework theoretically allows us to construct an arbitrary (according to limited resource) randomness test over a set of chosen operations. Therefore it can be viewed as a general framework for the test construction and should (hypothetically) provide a better detection ability than standard tests. Last but not least, a lower amount of data extracted from a given function is necessary to provide a working test.

This paper is organised as follows: Section 2 is devoted to randomness testing and general description of standard randomness tests. A reader familiar with these issues can proceed straight to the following section. In Section 3 we describe our new approach and compare it to the previous one. Section 4 describes testing procedures and experiment results obtained from testing the chosen eSTREAM candidates. Section 5 concludes the paper. The Appendix describes generalised empirical tests of randomness and used Goodness-of-Fit tests. The Appendix also describes settings used in our experiments and testing of both the model and its implementation.

1.1 Previous work

Knuth described basic simple tests of randomness in the second book (Knuth, 1997) of his well-known series. These tests form the basic randomness testing procedures and have been implemented (at least some of them) in all known test batteries. Marsaglia with Tsang proposed tests (Marsaglia and Tsang, 2002) that belong to the strongest that have been proposed so far. These tests were implemented in the Diehard battery (Marsaglia, 1995). The Diehard battery, as well as its newer version called Dieharder (Brown, 2004),

is focused on testing sequences of random numbers. It consists of 20 powerful tests that work with small sample sizes padded to 32 bits. The Statistical test suite developed by NIST (Rukhin, 2010) was proposed for testing and certification of random number generators used in cryptographic applications. It consists of 15 tests with a small overlap with Diehard. The Diehard battery along with the NIST STS are the most popular tools for randomness testing.

In 2007 Lecleuyer and Simard introduced quite a strong test battery called TestU01 (L'Ecuyer and Simard, 2007). TestU01 resolves the problems of Diehard and implements a larger variety of tests than any other battery. Other test suites exist, yet they are either commercial (Crypt-X (W. Caelli, 1992)) or provide only basic testing (ENT (Walke, 1993)). Earlier work (Švenda et al., 2013) comes with the approach of distinguisher construction that we use as the grounding concept for our general framework presented in this paper.

1.2 Previous approach

In the work (Švenda et al., 2013) the authors looked for a distinguisher of a bitstream produced by a cryptographic function (stream cipher) and a truly random bitstream (reference bitstream) produced by a physical source of randomness (quantum random generator (Stevanovi, 2007)). The implementation is available as an open-source project EACirc (Švenda and Ukrop, 2013).

Their distinguisher is constructed as a circuit-like software from a predefined set of operations. The circuit-like software is a small program that simulates a standard hardware circuit. It consists of wires and gates (operations) grouped into layers. The most important fact is that the functionality of the circuit (circuit-like software) can be simply changed by replacing operations in gates or by redirection of wires. This property is used for an iterative construction of distinguishers. The construction is controlled by a genetic algorithm (GA) that uses the success rate (percentage of correct answers) of distinguisher as its fitness value.

Using this approach, the authors obtained results that are somewhat comparable to those obtained by the NIST STS battery. The main problem of the previous approach is the interpretation of results. To evaluate randomness, the authors compare the computed average success rate of circuits with its reference value obtained by distinguishing two truly random bitstreams. When the reference average success rates are significantly different, it is evident that the distinguishing really works and circuits can be declared as real distinguishers. For close rates, it is hard to decide whether a small difference of success rates was caused by GA and its stochasticity or a weak distinguisher was found. In the new approach we give clear statistical interpretation of results and we magnify sensitivity of randomness test.

2 Randomness testing

Empirical tests of randomness fall under the standard statistical model – statistical hypothesis testing. Tests formulate a H_0 hypothesis "the bitstream is random" and an alternative hypothesis "the bitstream is not random". Each randomness test is defined by the test statistic S, which is a real-valued function of a numeric sequence. Tests are evaluated by comparing the P-value (computed from the test statistic value) with a chosen significance level α. For the P-value computation, it is necessary to know an exact distribution of S under hypothesis H_0 or at least its close approximation.

2.1 Standard tests of randomness

All standard tests of randomness are defined by the test statistic S carefully chosen to minimize the probability (β) of the Type II error (acceptance of H_0 for a non-random bitstream) for a fixed significance level α (probability of Type I error, rejection of H_0 for a random bitstream).

Note 1. *It should be noted that we do not construct standard empirical tests of randomness since S is constructed randomly and Type II error is not minimized.*

A P-value is computed from the "observed" test statistic a value s_{obs} using a theoretical distribution of test statistic values under the H_0 hypothesis. This reference distribution is determined by mathematical methods. It represents the distribution of test statistic values for random bitstreams. The significance level of standard tests is usually set to $\alpha = 1\%$. This means that standard tests accept the H_0 hypothesis if the P-value is greater than $\alpha = 0.01$. In such case we conclude that the examined bitstream is random with respect to the analysed feature.

Note 2. *Besides acceptance (pass) and rejection (fail) of the H_0 hypothesis, Diehard tests can also give a third result – weak. Diehard tests provide two-tailed testing with the treshold $\alpha = 0.5\%$ for weak data and 0.1% for non-random data (failed).*

Each standard test of randomness can be performed using the χ^2 test (Sheskin, 2003). It suffices to apply an appropriate categorisation function C to

the analysed bitstream and to compare the obtained results with the expected results. Standard tests of randomness can be implemented using an appropriate categorisation function C in the following steps:

1. Computation of observed frequencies: According to the purpose of the test, apply the function C to an adequate blocks of the bitstream B. Based on the function result, categorise each block and compute (observed) frequencies for each category.
2. Expected frequencies estimation: Estimate exact probabilities for each category if C would be applied to blocks of an infinite truly random bitstream. Using estimated probabilities, compute corresponding expected frequencies for a finite bitstream of a given length (length of B).
3. Evaluation: Use the χ^2 test to compare observed and expected frequencies.

More detailed explanation and a general description of standard tests can be found in Section B of the Appendix.

Each standard randomness test in the equivalent χ^2 form is fully defined by the function C and by this function is applied to the bitstream. In order to describe the general test we need to describe the general function C. In fact, we need to answer the following questions:

- Can the categorisation function C be described generally?
- How to apply function C to the bitstream?

To answer these questions, let us have a look at standard tests. In the BlockFrequency and Monobit tests different blocks of a fixed length are mapped to disjoint categories. In more complicated tests, such as Rank test, Linear Complexity test or Spectral test, the function C transforms the bitstream (or its parts) into other structures (matrix, LFSR, etc.) and after that it categorises them. Evidently, there is no simple generalisation of the C function functionality. On the other hand, we can identify several common properties of functions used in standard tests. All functions C process blocks of bits of a fixed length. Moreover, the number of resulting categories is small and therefore C can be simulated by a function with a small output. We have two situations with respect to the input size. Processed blocks are either quite long (1/100 of the length of B) or very small (several bits).

To answer the second question we made an observation that C is usually applied to consecutive non-overlapping blocks of the bitstream (original or transformed).

3 NEW APPROACH

Our goal is to construct simple tests of randomness that indicate a given bitstream to be non-random with a high confidence (small P-value). We aim to construct randomness tests in their general form (see Section B of the Appendix) defined by a categorisation function C. We assume functions C with a small input size (up to 1000 bits) that are applied to non-overlapping blocks of the bitstream. Since tests can be also viewed as distinguishers from random bitstreams, we can use an implementation (EACirc project (Švenda and Ukrop, 2013)) of the previous approach (Švenda et al., 2013). In fact, we try to improve the previous approach and to give a clear statistical interpretation of its results. Our new approach is based on ideas and principles used in standard empirical tests of randomness combined with the distinguisher construction.

3.1 Test construction

In the previous approach, the authors looked for distinguishers represented by circuit-like software while in the new approach, we look for distinguishers based on empirical tests of randomness. For this purpose, we have slightly modified the previous implementation. In the new implementation (EACirc2), the circuit C' represents a categorisation function C that defines the test of randomness. The new implementation could be divided into two modules: The first module, genetic algorithm module (GA), controls the construction of randomness tests (categorisation functions C). This module was taken from the previous approach without changes. The new second module, test of randomness module (TR), undertakes the testing of randomness based on the categorisation function C represented by the circuit. The GA module controls the evolution of circuits C'. The TR module uses circuits C' (categorisation function) for computation of fitness values for GA. Both modules work with circuits C' with fixed size input (n bits) and output (m bits).

Test construction is iterative and one iteration can be described by the following pseudocode:

1. The GA module sends evolved categorisation functions C (circuits C') to the TR module.
2. The TR module uses C' to test randomness of a given bitstream (its part) and sends computed P-values back to the GA module.
3. The GA module takes the P-values and uses them as fitness values for evolving of the next generation of circuits.

The GA controls evolution of functions C in order to minimize P-values.

3.2 TR module

In the TR module, the circuit C' is used for randomness testing according to the general testing procedure described more precisely in Section B of the Appendix. The testing procedure consists of the following three steps:

1. computation of observed frequencies,
2. estimation of expected frequencies,
3. evaluation.

In the first step, we apply the circuit C' to the analysed bitstream and we obtain observed frequencies (a histogram of results). In the second step, we compute the exact expected (theoretical) frequencies. The prediction of exact frequencies is for a general circuit C' without uniform distribution of the output a hard task. Therefore we use an estimation of the expected frequencies instead. A close estimate of expected frequencies can be obtained if we apply C' to a sufficiently long truly random bitstream generated by a physical source of randomness. In the third step, we use one sample Pearson's χ^2 test (Sheskin, 2003) to compare observed and expected frequencies and to compute the P-value of the test.

We have undertaken basic experiments and realised that for a sufficiently close estimate of expected frequencies we need to apply C' to a very long random bitstream. However, for practical reasons, C' is applied only to a short bitstream. To solve problems occurring from inaccurate approximation of expected frequencies, obtained and expected frequencies can be compared using a two-sample test. In our approach, we use the two-sample χ^2 test from (NIST, 1993) since the distribution of test statistic values for two-sample χ^2 tests is also the χ^2 distribution.

Now we describe the final version of the TR module in more detail. Let us assume that circuit C' has n bits of input and an m-bit output. In the TR module, C' is applied to non-overlapping blocks of the size n. In the practice, we divide the bitstream B to blocks B_i, called test vectors of the length of n bits. In fact, the TR module processes two bitstreams. The first bitstream (B) is the bitstream we want to test for randomness. The second (reference) bitstream R (produced by a physical source of randomness) is used for computation of expected frequencies. For simplicity, we assume bitstreams R, B of the same length. Thus corresponding sets of test vectors R_i, B_i have the same size denoted by k. The TR module computes the P-value for the test vectors $R_j, B_j, j \in \{1, \cdots, k\}$ as follows:

1. Computation of observed frequencies: The categorisation function C represented by the circuit C' is applied to all test vectors $B_j, j \in \{1, \cdots, k\}$. Frequencies o_i for each category (defined by resulting value O_i of C) are computed using $o_i = |\{B_j, j \in \{1, \cdots, k\} : C(B_j) = O_i, \}|$.
2. Expected frequencies estimation: Expected frequencies are computed by the same way from the test vectors R_i of the truly random bitstream, i.e., $e_i = |\{R_j, j \in \{1, \cdots, k\} : C(R_j) = O_i\}|$.
3. Evaluation: Two-sample χ^2 test is used to compute the test statistic value defined as

$$s_{obs} = \sum_{i=1}^{2^m} \frac{(o_i - e_i)^2}{o_i + e_i}.$$

3.3 GA module

As was stated before, we used GA module from previous EACirc project (Švenda and Ukrop, 2013). In this section we describe GA on general level only. More details about GA module can be found in (Švenda et al., 2013).

The GA module controls the evolution of circuits C' based on fitness values (P-values) computed in the TR module. In the GA module, test vectors are changed in order to prevent overlearning of circuits to a specific bitstream. The evolution consists of two phases: learning and testing. These two phases periodically alternate during the evolution. In the learning phase, test vectors are fixed and circuits are evaluated according to them. In the testing phase, the test vectors are changed and the TR module computes P-values from these new test vectors. The learning phase lasts for a fixed number of iterations (generations). The testing phase lasts a single iteration. This iteration starts a new learning phase with new test vectors.

More precisely, let f (frequency of changing test vectors) denotes the number of generations in each learning phase, then test vectors are changed in each lf-th iteration (population of circuits) for some integer l. Thus learning phases start in lf-th iteration and end in the $((l+1)f-1)$-th iteration. Testing phases are performed in the iterations if for some integer i.

3.4 Results and their interpretation

Results of our approach consist of series of P-values computed during the whole evolution (e.g., 30000 generations). The interpretation of computed

P-values is based on the fact that for a true H_0 hypothesis P-values are uniformly distributed in the interval $[0,1]$. This is, in fact, true for all statistical tests as well as for the tests of randomness.

Of course, the basic criterion of statistical testing must be fulfilled: "Analysed data must be taken from a random sample." In our case, this criterion can be interpreted as independence between the circuits (statistical tests) and test vectors. Since circuits are evolved according to test vectors, test vectors are independent from circuits only in the testing phases, where test vectors just changed. For clear interpretation we can use only one P-value computed in the testing phase. The reason is that circuits are partially correlated and thus P-values computed from a given set of test vectors are also correlated. Let P_i denotes set of all P-values computed in i-th generation. During the evolution sets $\mathbb{P}_1, \mathbb{P}_2, \cdots, \mathbb{P}_{30000}$ of P-values are computed. Let $\mathcal{P}$ denotes the set of P-values $\mathcal{P} = \{P_1, P_2, \cdots, P_{30000/f}\}$, where each P-value $P_i \in \mathbb{P}_{if}$ is taken from different testing phase (iteration if). In the case of a true H_0 hypothesis each P_i is taken from uniformly distributed set $\mathbb{P}_{if}$ on the interval $[0,1]$. In such case all P-values in the set $\mathcal{P}$ are also uniformly distributed on the interval $[0,1]$.

For testing uniformity of P-values, we use the Kolmogorov-Smirnov (KS) test described in Section A of the Appendix. Of course, KS computes its own P-value that can be compared to the significance level (chosen as $\alpha = 5\%$) to evaluate the KS test. Since P-values computed by the KS test could be smaller than α even for uniformly distributed P-values, we prefer to repeat the whole process several (r) times. For a proper and clear statistical interpretation we test whether 5% of P-values computed by the KS test from different sets $\mathcal{P}$ are smaller than the chosen significance level $\alpha = 5\%$. In practice, we do not compute the P-value of the KS test, but use another approach. We compare the KS statistic value with the critical statistic value $D = \frac{1.36}{\sqrt{t}}$ computed for $\alpha = 5\%$ (see Section A.3 of the Appendix) and for the set $\mathcal{P} = \{P_1, \cdots, P_t\}$ where $t > 35$.

4 Experiments

In this section we describe settings and results of performed experiments. We used our approach for randomness testing of bitstreams produced by selected stream ciphers from the eSTREAM project. Truly random data used for test vectors were produced by the Quantum Random Bit Generator Service (Stevanovi, 2007). We did all experiments with the new open-source implementation of EACirc2 project that can be found with previous EACirc and other testing tools and under following link (Švenda and Ukrop, 2013).

For expermiments we used following parameters of GA module:

1. circuit resources:
   - input length n = 128 bits, output length = 8 bits,
   - number of layers = 4, maximum number of connectors to gate = 4,
   - set of operations = Byte XOR, AND, NOR, NAND, NOT;
2. GA setting:
   - population size = 1, crossover probability = 0, mutation probability = 0.05,
   - number of generations gen = 30000, frequency of changing test vectors f = 100, number of test vectors k = 500,
   - fitness value = P-value of the test C' applied to test vectors.

Most of the parameter values were taken from the previous work (Švenda et al., 2013) in order to compare our new approach with the previous one. It should be noted that results strongly depend on the character of examined data and therefore it is quite difficult to find an optimal setting. However, sensitivity of the tests is given by resources of the circuits. In general, the more resources (larger input/output length, more layers, etc.) means better results. On the other hand, it also means more time needed for execution. Parameters such as output length, set of operations and fitness function were changed to improve the detecting ability of tests and to get results in reasonable time. More details about the settings can be found in Section C of the Appendix.

Before testing stream ciphers, we performed experiments that confirmed correctness of the statistical model and its implementation. More details about the implementation and model testing can be found in Section D of the Appendix. We tested a reference situation – randomness testing of a random bitstream. The model expects that a set of 300 (gen/f) P-values is uniformly distributed on the interval $[0,1]$ for each run of the algorithm. We performed 1000 runs, 49 out of which failed the KS test for uniformity. This is in a good agreement with the statistical model since 4.9%(=49/1000) is almost identical to expected value $\alpha = 5\%$ of the KS test.

After testing the model and its implementation we used our approach for randomness analysis of bitstreams produced by stream ciphers Grain, Decim, Fubuki, Hermes, LEX, Salsa20, TSC9 and Hermes with a limited number of rounds. Results of our

approach (EACirc2) are summarised in tables together with results of previous approach (EACirc), Dieharder battery (version 3.31.1) and NIST STS (version 2.1). Results of the Dieharder battery, NIST STS and previous approach are taken from (Švenda et al., 2013).

Each cell in the Dieharder battery or NIST STS column represents the number of tests that detected non-randomness in the given bitstream. Since Dieharder tests provides three levels of evaluation (pass, weak, fail, see NOTE 2 in section 2.1), values 1, 0.5, 0 respectively, were assigned to these levels and sum over all tests was taken. There were 20 tests from the Dieharder battery and 162 tests (tests with different parameters (Švenda et al., 2013)) from NIST STS with chosen significance level $\alpha = 1\%$ for both batteries. Cells in the EACirc column represent average success rates (percentage) of constructed distinguishers with 52% as the reference value obtained when distinguishing between two random bitstreams. Cells in the EACirc2 column represent percentage of runs for which the set of the *P*-values failed the KS test for uniformity with the significance level $\alpha = 5\%$ as the reference value (theoretical and practical). Note that values in different columns are not directly comparable and must be compared to different reference values. These reference values (0, 0, 52, 5) of particular (Dieharder, NIST, EACirc, EACirc2) columns are obtained from tests performed over the random sequence. We will get values (20,162,100,100) in the case of a 'totally' non-random sequence.

Rounds	Dieharder	NIST	EACirc	EACirc2
1	20.0	162	100	100
2	20.0	162	100	100
3	0.5	2	52	4

Table 1: Results for Grain.

Rounds	Dieharder	NIST	EACirc	EACirc2
1	20	162	100	100
2	19.5	162	54	100
3	19	162	53	100
4	16.5	83	52	100
5	15.5	83	52	92
6	1	4	52	5

Table 2: Results for Decim.

Rounds	Dieharder	NIST	EACirc	EACirc2
1	0	0	52	7
2	0	0	52	5

Table 3: Results for FUBUKI.

Rounds	Dieharder	NIST	EACirc	EACirc2
1	0	0	52	3
2	0	0	52	9

Table 4: Results for Hermes.

Rounds	Dieharder	NIST	EACirc	EACirc2
1	20	162	100	100
2	16	161	100	100
3	19.5	161	100	98
4	0	0	52	8

Table 5: Results for LEX.

Rounds	Dieharder	NIST	EACirc	EACirc2
1	14.5	161	87	100
2	14.5	161	87	100
3	0	0	52	4

Table 6: Results for Salsa20.

Rounds	Dieharder	NIST	EACirc	EACirc2
1-8	20	162	100	100
9	19	161	100	100
10	18	149	100	8
11	10	5	52	6
12	4	0	52	13
13	0	0	52	5

Table 7: Results for TSC.

Results in the EACirc2 column that were obtained from 100 runs (repetitions $r = 100$ of whole testing process) shows that our approach is significantly better than the previous approach EACirc (Švenda et al., 2013). While the Dieharder battery is still a slightly better tool for randomness testing compared to our approach, we achieve results comparable to those of NIST STS. Results of testing FUBUKI with 1 round, Hermes with 2 rounds, LEX with 4 rounds and TSC with 12 rounds indicate that EACirc2 detects non-randomness in some cases where NIST STS and Dieharder batteries, and previous approach (EACirc), fail. We tested the above-mentioned ciphers again with an additionally increased number of runs (r = 1000) in order to confirm previous results. We obtained results: 6.4% for FUBUKI, 6.4% for Hermes, 5.9% for LEX and 4.3% for TSC. Results confirm that our approach clearly detected non-randomness for Hermes with 2 rounds and FUBUKI with 1 round.

5 Conclusion

We have proposed a general design for the construction of empirical tests of randomness. Our new ap-

proach is based on work (Švenda et al., 2013) and improves it in two ways. It gives a clear statistical interpretation of its results and improves efficiency and success rate of the distinguisher (test of randomness) construction. We have tested our approach on several stream ciphers (with a reduced number of rounds) taken from the eSTREAM competition. Obtained results imply that our approach provides a significantly stronger randomness analysis than the previous one. Moreover, we have been able to detect non-randomness in some bitstreams (Hermes reduced to 2 rounds, FUBUKI reduced to 1 round) that fully pass standard batteries Dieharder and NIST STS. While the Dieharder battery is still a slightly better tool for randomness testing than our approach, we achieve results comparable to those of the NIST STS battery. Our future work will cover randomness testing of eSTREAM ciphers and SHA-3 candidates with various settings of our approach.

Acknowledgement

The first author was supported by the Ministry of Education, Youth, and Sport project CZ.1.07/2.3.00/30.0037 – Employment of Best Young Scientists for International Cooperation Empowerment. Other authors were supported by the Czech Science Foundation, project GAP202/11/0422.

REFERENCES

Brown, R. G. (2004). Dieharder: A random number test suite, version 3.31.1.

Knuth, D. E. (1997). *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

L'Ecuyer, P. and Simard, R. (2007). TestU01: A C library for empirical testing of random number generators. *ACM Trans. Math. Softw.*, 33(4).

Marsaglia, G. (1995). The marsaglia random number CDROM including the diehard battery of tests of randomness. http://www.stat.fsu.edu/pub/diehard/.

Marsaglia, G. and Tsang, W. W. (2002). Some difficult-to-pass tests of randomness. *Journal of Statistical Software*, 7(3):1–9.

NIST (1993). Two-sample χ^2 test. http://www.itl.nist.gov/div898/software/dataplot/refman1/auxillar/chi2samp.htm.

Rukhin, A. (2010). A statistical test suite for the validation of random number generators and pseudo random number generators for cryptographic applications, version STS-2.1. *NIST Special Publication 800-22rev1a*.

Sheskin, D. J. (2003). *Handbook of parametric and nonparametric statistical procedures*. crc Press.

Stevanovi, R. (2007). Quantum random bit generator service. http://random.irb.hr/.

Švenda, P. and Ukrop, M. (2013). EACirc project, https://github.com/petrs/eacirc.

Švenda, P., Ukrop, M., and Matyáš, V. (2013). Towards cryptographic function distinguishers with evolutionary circuits. In *SECRYPT*, pages 135–146. SciTePress.

W. Caelli, e. a. (1992). CryptX package documentation. Technical report, Information Security Research Centre and School of Mathematics, Queensland University of Technology. http://www.isrc.qut.edu.au/resource/cryptx/.

Walke, J. (1993). Ent - a pseudorandom number sequence test program. http://www.fourmilab.ch/random/.

Zhang and Jin. Incomplete gamma function. http://www.crbond.com/math.htm.

APPENDIX

A Goodness-of-Fit tests

The family of Goodness-of-Fit tests can be formally divided into two main classes. According to the number of analysed samples we talk about one-sample tests or two-sample tests. The one sample Goodness-of-Fit test measures how well a given sample fits a statistical model (expected distribution). The two-sample test analyses whether two samples came from the same distribution. Two of the most frequently used statistical tests are Pearson's χ^2 test (Sheskin, 2003) and Kolmogorov-Smirnov (KS) test (Sheskin, 2003).

A.1 χ^2 test

The χ^2 test is typically used for testing whether the observed frequency distribution fits the theoretical distribution. This test is applied to categorised (binned) data and test statistics depended on the data categorisation. In the test we assume that observations of some events fall into k mutually exclusive categories. The one sample χ^2 test statistic is defined as

$$\chi^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i},$$

where o_i denotes the frequency in the i-th cathegory. The P-value of the test is computed using the refer-

ence χ^2 distribution that is determined by the degree of freedom $(k-1)$.

Note 3. *In practice, P-value is computed from χ^2 test statistic by the gamma function and the incomplete gamma function. We tested several open-source implementations of the incomplete gamma function, but most of them are inaccurate for extremal arguments (close to zero, greater than 200). We use the implementation of the incomplete gamma function from (Zhang and Jin,) that produces correct values for arbitrary arguments.*

The χ^2 test uses distribution with k categories that closely approximates binary distributed variable ($k = 2$) or multinomially distributed variable ($k > 2$). For a sufficient approximation of a multinomial variable, frequency in each category should be greater than 5, i.e., $e_i \geq 5$ for all $i \in \{1, \cdots, k\}$.

A.2 Two-sample χ^2 test

In our approach we use a two-sample χ^2 test (NIST, 1993) that compares the distributions of two samples since expected frequencies e_i can not be approximated closely. The test statistic for the two-sample χ^2 test changes to $\chi^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i + o_i}$. The reference distribution is also the χ^2 distribution with $d-1$ degrees of freedom. In the two-sample χ^2 test, d represents the number of non-empty categories $(o_i + e_i > 0)$. Moreover, for a correct analysis using the two-sample χ^2 test, it is required that the sum $o_i + e_i$ should be at least 5. Thus only categories with $o_i + e_i \geq 5$ are used for computation of the test statistic value.

A.3 Kolmogorov-Smirnov test

The one sample Kolmogorov-Smirnov (KS) (Sheskin, 2003) test is a more universal Goodness-of-Fit test than the χ^2 test since it can be used also for testing continuous distributions. The KS test compares an empirical distribution of the sample with the reference distribution using the cumulative distribution functions (CDF). Let $\bar{F}(x)$ denotes an empirical CDF, then $\bar{F}(x)$ is defined as $\bar{F}(x) = Pr(X < x)$. The KS test statistic D is defined by $D = sup_{x \in R} |\bar{F}(x) - F(x)|$, where $F(x)$ denotes the expected value of the cumulative distribution. We use the KS test for testing the uniformity of P-values (P_i , $i \in \{1, 2, \cdots, t\}$) on the interval $[0, 1]$. In a such discrete case the CDF has the form

$$\bar{F}(t) = \frac{\#i : P_i < x}{t}.$$

For sorted P-values $P_1 \leq P_2 \leq \cdots \leq P_t$ we can write $F_t(x) = i$ iff $P_i \leq x$ and $P_{i+1} > x$. Reference CDF is defined on the interval of our interest as $F(x) = x$. The KS test statistic D can be computed as $D = \max_{i=1}^{t-1}(\max(|\frac{i}{t} - P_i|, |\frac{i+1}{t} - P_i|))$. For our computations we use the critical value D_α of the significance level $\alpha = 0.05$ that can be computed as $D_{0.05} = \frac{1.36}{\sqrt{t}}$ for $t > 35$.

B General description of standard tests

For better understanding of general tests, we first describe a standard test of randomness. In the standard tests, statistic S is applied to the bitstream to obtain the test statistic value s_{obs}. This value is used for computation of the corresponding P-value. To clarify the testing procedure, let us take the simplest randomness test, the Monobit test from NIST STS (Rukhin, 2010).

Example 1. *The Monobit test looks for irregularities in the proportion of counts of ones and zeros in a sequence. The Monobit test is defined by the test statistic $S = \frac{S_n}{\sqrt{n}}$. The value S_n is computed as an absolute value of difference of count of ones and zeros in the bitstream of the length n. The value S_n is computed as $S_n = |\sum_{i=1}^{n} \varepsilon_i|$, for $\varepsilon_i \in \{1, -1\}$ (zero bits converted to -1). The reference distribution (under H_0 hypothesis) of the test statistic (for large n) is half normal. The test uses $erfc()$ function for computation of the P-value from normal distributed $\frac{s_{obs}}{\sqrt{2}}$.*

Let us assume a bitstream $\varepsilon = 1011010101$ with the length $n = 10$. Using the previous expressions, we can compute $S_n = 2$ and the statistic test value $s_{obs} = \frac{2}{\sqrt{10}}$. To compute the P-value we apply the $erfc()$ function to $\frac{s_{obs}}{\sqrt{2}}$ and we finally get

$$P - value = erfc\left(\frac{2}{\sqrt{10}}\right) = 0.52708926.$$

Evaluation of the H_0 hypothesis from the previous example can be also done using other appropriate statistical tests. The Monobit test of randomness checks whether the frequencies o_i of zeros (o_0) and ones (o_1) in a sequence fit to the expected theoretical frequencies e_i. We can therefore use any one sample Goodness-of-Fit test such as the χ^2 test or the KS test.

The Monobit test for the bitstream of the length n can be evaluated by the χ^2 test as follows:

1. Computation of observed frequencies: take each bit of the analysed bitstream, categorise it according to its value (0 or 1) and compute the frequency for each category $(o_0 = 4, o_1 = 6)$.
2. Expected frequencies estimation: estimate probabilities of bits for a truly random infinite bitstream ($p_0 = p_1 = 0.5$ for zeros and ones) and use them to compute the expected frequencies for a bitstream of the given length $(e_0 = e_1 = 5)$.

3. Evaluation: compare observed and expected frequencies by the χ^2 test with 1 degree of freedom (P-value = 0.52708926).

In fact, these steps can be generalized for all other empirical tests of randomness.

The categorisation function corresponds to the purpose of the test and can be deduced directly from its test statistic. In the case of the Monobit test, the categorisation function was applied to individual bits, but in general it can process parts of the examined bitstream or its transformed equivalent. In general, we also assume that the number of categories is arbitrary (not only two), and so is the related degree of freedom.

C Parameters and settings

Size of the population was changed to one, since only one P-value can be taken for the correct statistical interpretation (see Section 3.4). This compensates for the fact that the whole process must be repeated several times (for the correct interpretation of results). The fitness value of the GA was changed to the P-value since the P-value clearly represents the most relevant value of the testing procedure.

Note 4. *It is possible to use two-sample χ^2 test statistic as the fitness value but it does not reflect degrees of freedom.*

Since the population consists of only one circuit, the crossover probability is automatically set to zero. To choose appropriate circuit settings, a deeper analysis of the evolution is needed. Let us consider a fixed setting of the circuit parameters (number of layers, number of gates in the layer, ...). Since the circuit resources are limited, they should be used effectively. This means that the pool of available operations should consist of complex operations. The reason for that is that complex operations constructed from trivial ones consume a lot of available resources. On the other hand, too many defined operations significantly enlarge the space where GA works and could mislead the evolution process. Therefore the set of operations should consist of complex but meaningful operations. In the case of stream cipher, operations used in the cipher design can be considered meaningful. Therefore we use only simple Byte "boolean" operations like AND, OR, NOR, NOT, etc.

The main problem of the previous approach is that the output from the last layer of the circuit is interpreted as a single bit. Clearly, this leads to a loss of distinguishing ability of circuits, since results of many gates are often discarded. To avoid this, more bits from the last layer should be used for the interpretation. This perfectly fits to our framework since the categorisation function of the test can work with arbitrary many categories. It can be expected that we get the most sensitive test of randomness if all categories are defined by single output value of the circuit, i.e., if C with 2^m categories is represented by a circuit C' with m output bits. Unfortunately, application of the χ^2 test (Section A of this Appendix) requires that the frequency in each category should be at least 5. This means that for C' with 8 output bits there should be either more test vectors (more than used $k = 1000$) or the number of categories should be smaller. We have reduced the number of categories. For 1000 test vectors, it must be smaller than 200. In such case each category could be defined by 7 bits of a circuit output byte. Of course, the GA does not fill the categories evenly and therefore we need to use less categories. For our experiments we have chosen 8 categories defined by the last 3 bits of all 8 output bits.

D Implementation and model testing

In this part we describe tests that confirm correctness of the statistical model and correctness of its implementation. We want to confirm that P-values computed by two-sample χ^2 (from category frequencies) are distributed uniformly on the interval $[0,1]$. Besides the statistical model we also need to check our implementation of statistical tests (two-sample χ^2 test, KS test).

Firstly, we have tested the implementation of the KS test. We analysed 10^7 sets $\mathcal{P}$ of $t = 300$ uniformly distributed randomly generated real numbers from the interval $[0,1]$. Using the KS test we have obtained the total of 497496 test statistics values that were located in the critical region defined by $\alpha = 0.05$. This value represents 4.97% of all tested sets and is in good agreement with the expected 5% value.

Secondly, to check the χ^2 test implementation we simulate the circuit generation process. We generated 300 pairs of number samples $S_{i,1}, S_{i,2}, i \in \{1, \cdots, 300\}$ from the set $\{0, \cdots, 7\}$. For each i, both sets $S_{i,1}, S_{i,2}$ were randomly generated according to the given random distribution. We applied the two-sample χ^2 test to compare samples $S_{i,1}, S_{i,2}$ for each $i \in \{1, \cdots, 300\}$. We obtained the set $\mathcal{P}$ of 300 P-values P_i. This set was tested by KS test for uniformity on the $[0,1]$. We repeated the whole process 10^4 times (runs) and realised that 5.1% of KS test statistic values were located in the 5% critical region. This observation is also in a good agreement with the model. All performed tests indicate that the statistical model, our implementations of the KS test and the two-sample χ^2 test are correct.