The automated testing of randomness with multiple statistical batteries

Ľubomír Obrátil
lubomir.obratil@gmail.com

22. 6. 2017

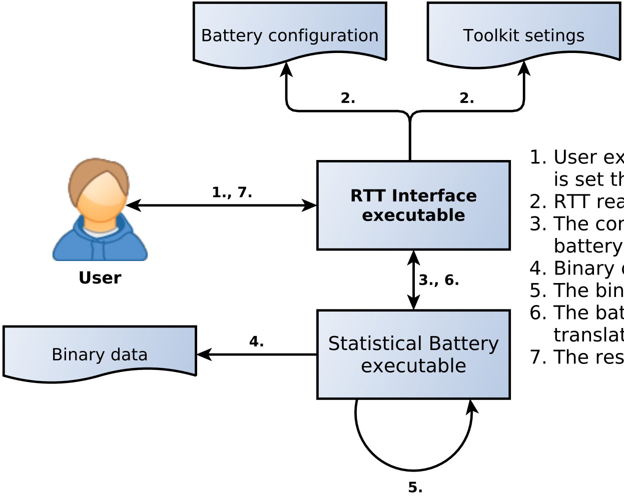
Thesis structure

- Creation of a unified interface supporting multiple statistical batteries.
    - NIST Statistical Testing Suite
    - Dieharder
    - TestU01
- Conducting the baseline (control) experiment to create a reference point for the further experiments.
- Evaluating randomness of outputs of well-known cryptographic primitives.
- Analysing validity of the Dieharder battery results.

Randomness Testing Toolkit – overview

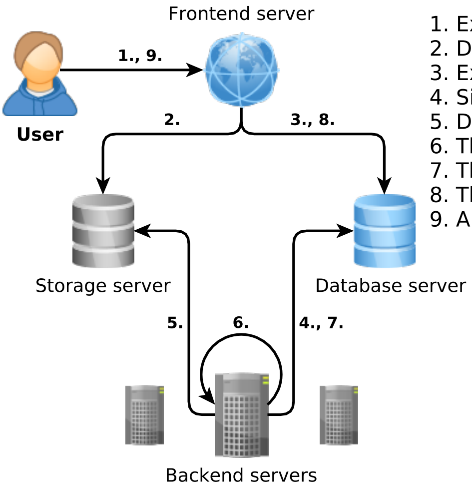
- Design and implementation of a tool for consistent randomness evaluation.
- Intended to be used by users without prior knowledge about statistical testing as well as by researchers in CRoCS.
- The developed tool (RTT) acts as a interface between the user and the statistical batteries – common format of the battery settings and results.
- The toolkit supports multiple statistical batteries – NIST STS, Dieharder and TestU01; it is possible to add more batteries over time.
- Both standalone program and online service were developed.

Randomness Testing Toolkit – local interface



1. User executes the binary, basic configuration is set through command-line arguments.
2. RTT reads advanced settings in common format
3. The common settings are translated into battery-specific settings.
4. Binary data is read from file supplied to battery.
5. The binary stream is analysed.
6. The battery-specific results are gathered and translated into unified format by RTT.
7. The results are stored for user to inspect.

Randomness Testing Toolkit – web service



1. Experiment and data are submitted.
2. Data is uploaded.
3. Experiment and related jobs are created.
4. Single pending job is selected.
5. Data is downloaded.
6. The data is analyzed.
7. The results of the analysis are stored.
8. The results in the database are presented.
9. Analysis of the data is inspected.

Statistical testing of randomness – 1/2

Testing hypothesis – H_0
During the experiments, we evaluated the hypothesis that the analysed data were produced by a truly random generator. We denote the hypothesis as H_0 (null hypothesis).

Statistical battery
Software with the purpose of detecting biases in data stream; collection of statistical tests.

Statistical test
Single unit in a statistical battery, checks some property of the data; e.g. count of ones. Output of a test is the probability that the analysed data were produced by TRNG. Each test in a given battery will either fail (H_0 rejection) or pass (H_0 retainment).

Statistical testing of randomness – 2/2

Significance level – α
The significance level is set prior to the experiments (usually 0.001) and based on it, the null hypothesis is rejected or retained.

False positive (Type I error)
The false positive result is observed when H_0 holds true but it is rejected – stream produced by TRNG is evaluated as non random. Probability of Type I error is α.

False negative (Type II error)
The false negative result is observed when H_0 is false but it is not rejected – stream generated by biased generator is evaluated as random.

Establishing baseline results

- The result of a battery is obtained from the proportion of the failed tests in the battery (e.g. we consider the data biased if more than 2 out of 15 tests fail). However, even data generated by a perfect TRNG may fail some tests (false positives).
- To identify the maximum count of failed tests (assuming that the H_0 holds) we processed 8 TB of data generated by a quantum random generator (control experiment). Further experiments were interpreted based on these results.
- In order to make the result interpretation more straight-forward, results of certain tests were grouped together. As a consequence, the counts of failed tests are closer to expected (theoretical) numbers and therefore more easy to evaluate.

Analysis of well-known algoritms

- Analyse outputs of well-known cryptographic algorithms (AES, DES, RC4, etc.).
- Observe the security margins of the algorithms.
- Compare the results to other approach in CRoCS.

Analysed algorithms

- In total, 72 different data streams were analysed.
- The data streams were outputs from 16 distinct round-reduced cryptographic algorithms.
- The algorithms were chosed based on their popularity (AES, DES, RC4, ...) or their success in crypto competitions eSTREAM and SHA3 (Rabbit, Keccak, Grøstl, ...).

Analysis conditions

- Each datastream was 8GB long.
- The conditions of analysis were same as in the previous experiment.
- The interpretation of the result was based on the results of the baseline

Analysis of the results of Dieharder battery

- Examine behavior of Dieharder battery during truly random data analysis.
- Analyse the distribution of the results.

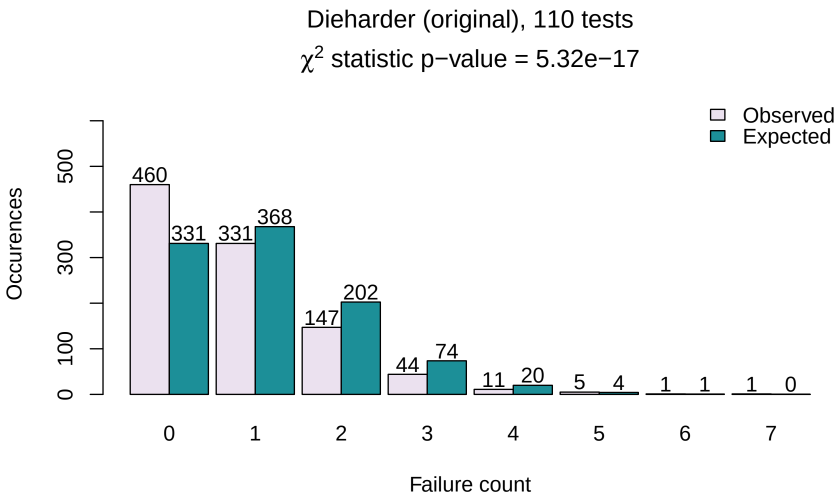
Usable testbed analysis – 2/3

Algorithm	Biased round RTT	Biased round EACirc	Security Margin
AES	3	3	7 – 70%
BLAKE	1	1	15 – 93.7%
Grain	6*	2	7 – 53.8%
Grøstl	2	2	12 – 85.7%
HC-128	–	–	0 – 100%
JH	6	6	36 – 85.7%
Keccak	3	2	21 – 87.5%
MD6	10*	8	94 – 90.3%
Rabbit	4*	0	0 – 0%
RC4	0*	–	0 – 0%
Salsa20	2	2	18 – 90%
SINGLE-DES	5	4	11 – 68.7%
Skein	4	3	68 – 94.4%
SOSEMANUK	4	4	21 – 84%
TEA	5	4	27 – 84.3%
TRIPLE-DES	3	2	13 – 81.2%

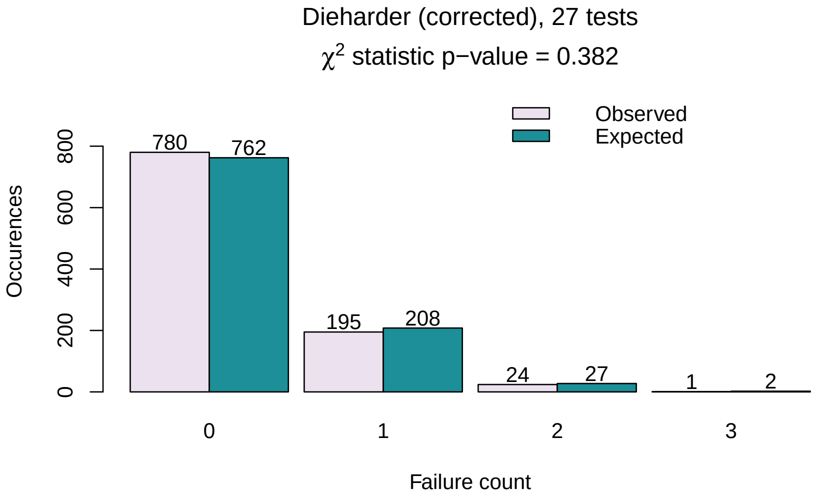
References

- **Randomness Testing Toolkit**
  https://github.com/crocs-muni/randomness-testing-toolkit
- **EACirc**
  https://github.com/crocs-muni/eacirc
- **NIST Statistical testing suite**
  http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html
- **Dieharder**
  http://www.phy.duke.edu/~rgb/General/dieharder.php
- **TestU01**
  http://simul.iro.umontreal.ca/testu01/tu01.html

Baseline experiment – uncorrected results



Baseline experiment – corrected results



Baseline experiment – resulting reference

	Closeness to expected results		
Battery name	Uncorrected	Uncorrected	**Allowed failures**
Dieharder	$5.32 \cdot 10^{-17}$	0.38	$3/27$
NIST STS	$2.17 \cdot 10^{-2}$	$4.44 \cdot 10^{-7}$	$2/15$
TU01 Small Crush	0.71	0.95	$2/10$
TU01 Crush	$3.36 \cdot 10^{-11}$	$3.31 \cdot 10^{-3}$	$3/32$
TU01 Rabbit	$2.02 \cdot 10^{-5}$	$1.45 \cdot 10^{-23}$	$2/16$
TU01 Alphabit	$2.14 \cdot 10^{-8}$	$2.8 \cdot 10^{-7}$	$1/4$
TU01 Block Alphabit	$1.87 \cdot 10^{-68}$	$5.15 \cdot 10^{-47}$	$1/4$

Usable testbed analysis – 3/3

Notable results

- **Grain** – Tests smarsa_MatrixRank and scomp_LinearComp (Crush, Rabbit) will fail in 3, 4, 5 and 6-round configuration.
- **MD6** – Tests smarsa_MatrixRank and sspectral_Fourier3 (Crush, Rabbit) will fail in 9 and 10-round configuration.
- **Rabbit** – Tests sstring_HammingIndep and sstring_PeriodsInStrings (Crush, Rabbit, Alphabit, Block Alphabit) will fail in **full** configuration.
- **RC4** – Tests sknuth_SimpPoker and sknuth_Gap (Crush) will fail in **full** configuration.

Analysis of Dieharder – 1/2

Analysed data

- 8TB of quantum random data processed continuously by the tests - single application of a test to a data stream will yield single first-level p-value
- Uniformity of the first level p-values was analysed.
- The p-values should be uniformly distributed on the interval $< 0, 1 >$.
- Total of 110 sets of p-values (single set per raw, uncorrected test) was inspected.
- Each set had a different size – usually between 1 to 2 millions of p-values per set.

Experiment results

- Out of 110 p-value sets, 39 sets were not uniformly distributed
- Chi-Square (χ^2) statistic used for uniformity testing. When the p-value of the statistic was less than 0.001, the inspected set was considered non-uniform.
- Flawed non-uniform distributions can have impact on Dieharder results.

Analysis of Dieharder – 2/2