Masarykova univerzita
Fakulta informatiky

Usage of evolvable circuit for statistical testing of randomness

Bachelor thesis

Martin Ukrop

Brno, spring 2013

Declaration

Hereby I declare, that this paper is my original authorial work, which I have worked out by my own. All sources, references and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Martin Ukrop

Advisor: RNDr. Petr Švenda, Ph.D.

Acknowledgement

Thanks will be here.

Abstract

Abstract will be here.

Keywords

Keywords will be here.

Contents

1 Introduction

Text ...

2 Statistical randomness testing

- idea: statistic (maths) -> test
- fast
- universal
- usage: assess quality of (pseudo)random data, ???

2.1 Statistical Test Suite by NIST

- nist standard
- short description (?)

2.2 Diehard battery of tests

- author
- one of the first and most-well known in those years
- old, but still considered "gold standard" along with sts-nist
- short description of tests (?)

2.3 Dieharder: A Random Number Test Suite

- framework idea
- progress: diehard -> sts-nist -> new

2.4 Limits and disadvantages of statistical testing

- idea -> test (idea is always the predecessor)
- check only one specific property
- can only rarely be adapted to specific situation
- results interpretation (what is wrong?)

3 Evolution based randomness testing

- general description of GA
- idea behind EACirc
- previous evolution of EACirc (SensorSim -> bc, mgr -> today)
- capabilities of EACirc
  - general object model (+picture)
  - separate modules for projects
  - separate modules for evaluators
  - guaranteed bit-reproducibility
  - computation recommencing (state, ...)
  - static checker for pregenerated test vectors
- EACirc is wider project beyond the scope of this thesis, thus project evolution, some parts are being redesigned

4 Experiment settings and results

- general evolution settings
- main goal: finding strong distinguisher (over 99% for 50 consecutive generations)
- displayed average stable generation across 30 independent runs
  (stable = fitness over 99% for at least next 50 test sets)
- if none stable generation was found, average average maximum fitness after test vector change is displayed in parentheses.
- statistical batteries STS-NIST and Dieharder for reference
- 250 MB of data used, same files for Dieharder and STS-NIST
- STS-NIST settings (lenghts, 2 test omitted)
- each test run 100 times on 1 000 000 bits
- some runs had problems with tests RandomExcursions and RandomExcursionsVariant, these tests are not included in the result
- STS-NIST results interpretation (scores 0, 1)
- Dieharder settings
- test corresponding to original Diehard (except for Diehard sums test)
- each test run once, length of the stream decided by test
- Dieharder results interpretation (scores 0, 0.5, 1)
- displayed number of tests passed out of total (pass=1, weak=0.5, fail=0)

5 Control distinguishers

- introduction – the need of reference numbers before analysis
- we need to define what does it mean "indistinguishable" in our setting
- we use quantum random data from Humboldt Universitat and Quantum random bit generator service as a standard for randomness

5.1 Looking for non-randomness in quantum random data

- trying to distinguish quantum random data from quantum random data
  => we presume to fail
- using random data from Quantum random bit generator service
- statistical batteries: data are random (Dieharder: 20/20, STS NIST: 188/188)
- evolution: no stable distinguisher found, AAM of 0.52 (differences in various runs in 3rd or 4th decimal place)
- presumption: dependence on test set size and population size
- presumption confirmed (Table 5.1), AAM decreases with smaller population and bigger test set size

		number of test vector in a set					
		200	500	1000	2000	5000	10 000
individuals in population	5	–	–	(0.509)	-	-	-
	10	–	–	(0.514)	-	-	-
	20	(0.544)	(0.527)	(0.520)	(0.514)	(0.509)	(0.506)
	50	-	-	(0.526)	-	-	-
	100	-	-	(0.530)	-	-	-

Table 5.1: Dependence of AAM on population size and test vector set size.

5.2 Distinguishing quantum random data from different sources

- distinguishing streams of quantum random data from Humboldt University and streams of quantum random data from Ruđer Bošković Institute
  - Quantum Random Bit Generator Service, Centre for Informatics and Computing, Ruđer Bošković Institute, Zagreb, Croatia
  - Quantum Random Number Generator Service, Department of Physics, Humboldt University, Berlin, Germany
- 6 files of 5 MB from each source
- fixed initial reading offsets as (0,0)
  => same data from given file in each run
- looking for distinguisher for each pair
- interpretation of results (Table 5.2):

    - data from both sources are equally random for our purposes
    - there is no single statistically different stream in these
      =>they can be used interchangeably

		QRBG service (Ruđer Bošković Institute, Croatia)					
		stream 1	stream 2	stream 3	stream 4	stream 5	stream 6
QRNG service (HU, Germany)	stream 1	(0.521)	(0.520)	(0.520)	(0.519)	(0.519)	(0.519)
	stream 2	(0.158)	(0.519)	(0.520)	(0.520)	(0.520)	(0.519)
	stream 3	(0.519)	(0.522)	(0.519)	(0.520)	(0.519)	(0.519)
	stream 4	(0.520)	(0.520)	(0.519)	(0.518)	(0.519)	(0.519)
	stream 5	(0.519)	(0.520)	(0.519)	(0.518)	(0.520)	(0.520)
	stream 6	(0.520)	(0.519)	(0.520)	(0.520)	(0.519)	(0.519)

Table 5.2: Distinguishing binary quantum random streams from independent sources.

5.3 Uncompressed audio streams

- 12 files
    - 3 quantum random files
    - 3 noise files (white, pink, brown)
    - 3 noise files with oscillating volume (2 cycles in given 30 seconds)
    - 3 samples of transcendental khaoblack metal music
- each file is 30sec of uncompressed WAV audio (5.3MB)
- evolving distinguisher for each pair
- interpretation of results (Table 5.3):
    - quantum random stream are undistinguishable (we already know)
    - random stream 2 is more "noise-like"
      => reminds us to use random data cautiously, and use statistics to evaluate results
    - uncompressed WAV audio is quite easily distinguishable from random stream of data (generally over 80%)
    - different types of noise can be quite successfully distinguished from one another (generally over 70%)
    - it is difficult to distinguish noise from its oscillating version (around 58%)
    - when using oscillating noise for distinguishing, fitness is not oscillating
      => volume is not statistically important in these sample noise files
    - given samples of metal music can be quite successfully distinguished from noise
    - given samples of metal music nearly cannot be distinguished from one another
- most of the runs have slow rising tendency in fitness
  => if more generations, the average maximum value might be slightly higher

		random streams			noise			noise (oscillating)			metal music		
		random stream 1	random stream 2	random stream 3	white noise	pink noise	brown noise	white noise (oscillating)	pink noise (oscillating)	brown noise (oscillating)	metal music (sample 1)	metal music (sample 2)	metal music (sample 3)
random	random stream 1	n/a	(0.52)	(0.52)	(0.91)	(0.96)	(0.97)	(0.87)	(0.93)	(0.95)	(0.79)	(0.84)	(0.88)
	random stream 2	(0.52)	n/a	(0.52)	(0.82)	(0.85)	(0.83)	(0.86)	(0.91)	(0.96)	(0.89)	(0.85)	(0.87)
	random stream 3	(0.52)	(0.52)	n/a	(0.94)	(0.96)	(0.95)	(0.95)	(0.96)	(0.91)	(0.82)	(0.88)	(0.87)
noise	white noise (constant)	(0.91)	(0.82)	(0.94)	n/a	(0.71)	(0.84)	(0.59)	(0.80)	(0.96)	(0.78)	(0.80)	(0.79)
	pink noise (constant)	(0.96)	(0.85)	(0.96)	(0.71)	n/a	(0.72)	(0.70)	(0.63)	(0.68)	(0.70)	(0.74)	(0.75)
	brown noise (constant)	(0.97)	(0.83)	(0.95)	(0.84)	(0.72)	n/a	(0.80)	(0.65)	(0.53)	(0.80)	(0.73)	(0.83)
noise (osc.)	white noise (oscillating)	(0.87)	(0.86)	(0.95)	(0.59)	(0.70)	(0.80)	n/a	(0.80)	(0.84)	(0.80)	(0.80)	(0.80)
	pink noise (oscillating)	(0.93)	(0.91)	(0.96)	(0.80)	(0.63)	(0.65)	(0.80)	n/a	(0.62)	(0.81)	(0.84)	(0.82)
	brown noise (oscillating)	(0.95)	(0.96)	(0.91)	(0.96)	(0.68)	(0.53)	(0.84)	(0.62)	n/a	(0.75)	(0.84)	(0.86)
metal music	metal music (sample 1)	(0.79)	(0.89)	(0.82)	(0.78)	(0.70)	(0.80)	(0.80)	(0.81)	(0.75)	n/a	(0.54)	(0.54)
	metal music (sample 2)	(0.84)	(0.85)	(0.88)	(0.80)	(0.74)	(0.73)	(0.80)	(0.84)	(0.84)	(0.54)	n/a	(0.57)
	metal music (sample 3)	(0.88)	(0.87)	(0.87)	(0.97)	(0.75)	(0.83)	(0.80)	(0.82)	(0.86)	(0.54)	(0.57)	n/a

Table 5.3: Distinguishing random streams and uncompressed audio (noise, oscillating noise, metal music).

6 Distinguishing cipher outputs from random stream

- introduction, idea, running EACirc along with statistical batteries
- stream ciphers from eStream competition

6.1 Stream ciphers used

- ciphers except for ?? (why??)
- from last phase
- those that could be limited in rounds are tested in weaker variant as well
- differences from Metej Pristak thesis

6.2 Generating binary stream from stream ciphers

- cipher modes (iv+key initialization frequency)
- case of LEX (not weakening the cipher, only making shorter output)
- case of TSC (producing binary stream of 0 for 1-8 rounds) => problems in 3 Dieharder tests

6.3 Results interpretation

- ???
- more or less as statistical batteries
- dieharder better in some case than STS-NIST (is newer and some tests are redesigned)
- statistical tests has much more input data compared to EACirc
- using evolved distinguisher is quick

# of rounds	IV and key reinitialization								
	once for run			for each test set			for each test vector		
	Dieharder (x/20)	Sts Nist (x/162)	EACirc (AAM)	Dieharder (x/20)	Sts Nist (x/162)	EACirc (AAM)	Dieharder (x/20)	Sts Nist (x/162)	EACirc (AAM)
1	0.0	0	$n = 2681$	0.0	0	(0.85)	0.0	5	$n = 1431$
2	0.5	0	(0.54)	1.0	0	(0.54)	15.5	146	(0.52)
3	1.0	0	(0.53)	1.0	0	(0.53)	15.0	160	(0.52)
4	3.5	79	(0.52)	3.0	78	(0.52)	20.0	160	(0.52)
5	4.5	79	(0.52)	3.5	91	(0.52)	17.5	161	(0.52)
6	19.0	158	(0.52)	19.0	159	(0.52)	18.0	162	(0.52)
7	18.5	162	(0.52)	19.0	161	(0.52)	20.0	161	(0.52)
8	20.0	162	(0.52)	20.0	159	(0.52)	19.0	161	(0.52)

Table 6.1: Random distinguishers for Decim ciphertext.

# of rounds	IV and key reinitialization								
	once for run			for each test set			for each test vector		
	Dieharder (x/20)	Sts Nist (x/162)	EACirc (AAM)	Dieharder (x/20)	Sts Nist (x/162)	EACirc (AAM)	Dieharder (x/20)	Sts Nist (x/162)	EACirc (AAM)
1	20.0	162	(0.52)	20.0	161	(0.52)	18.0	162	(0.52)
4	20.0	162	(0.52)	20.0	162	(0.52)	20.0	162	(0.52)

Table 6.2: Random distinguishers for FUBUKI ciphertext.

# of rounds	IV and key reinitialization								
	once for run			for each test set			for each test vector		
	Dieharder (x/20)	Sts Nist (x/162)	EACirc (AAM)	Dieharder (x/20)	Sts Nist (x/162)	EACirc (AAM)	Dieharder (x/20)	Sts Nist (x/162)	EACirc (AAM)
1	0.0	0	$n = 221$	0.0	0	(0.67)	18.5	162	(0.52)
2	0.0	0	$n = 471$	0.5	0	(0.66)	20.0	162	(0.52)
3	19.5	160	(0.52)	20.0	162	(0.52)	20.0	162	(0.52)
13	20.0	162	(0.52)	20.0	161	(0.52)	19.5	162	(0.52)

Table 6.3: Random distinguishers for Grain ciphertext.

# of rounds	IV and key reinitialization								
	once for run			for each test set			for each test vector		
	Dieharder (x/20)	Sts Nist (x/162)	EACirc (AAM)	Dieharder (x/20)	Sts Nist (x/162)	EACirc (AAM)	Dieharder (x/20)	Sts Nist (x/162)	EACirc (AAM)
1	20.0	162	(0.52)	20.0	162	(0.52)	20.0	162	(0.52)
10	20.0	160	(0.52)	20.0	162	(0.52)	20.0	162	(0.52)

Table 6.4: Random distinguishers for Hermes ciphertext.

# of rounds	IV and key reinitialization: once for run — Dieharder (x/20)	once for run — Sts Nist (x/162)	once for run — EACirc (AAM)	for each test set — Dieharder (x/20)	for each test set — Sts Nist (x/162)	for each test set — EACirc (AAM)	for each test vector — Dieharder (x/20)	for each test vector — Sts Nist (x/162)	for each test vector — EACirc (AAM)
1	0.0	0	$n = 148$	0.0	0	$n = 7274$	3.0	1	$n = 154$
2	4.0	1	$n = 221$	4.0	1	$n = 304$	3.5	1	$n = 254$
3	0.5	1	$n = 378$	3.5	1	$n = 491$	4.0	1	$n = 361$
4	20.0	162	(0.52)	19.5	162	(0.52)	20.0	161	(0.52)
10	19.5	162	(0.52)	19.5	160	(0.52)	20.0	160	(0.52)

Table 6.5: Random distinguishers for LEX ciphertext.

# of rounds	IV and key reinitialization: once for run — Dieharder (x/20)	once for run — Sts Nist (x/162)	once for run — EACirc (AAM)	for each test set — Dieharder (x/20)	for each test set — Sts Nist (x/162)	for each test set — EACirc (AAM)	for each test vector — Dieharder (x/20)	for each test vector — Sts Nist (x/162)	for each test vector — EACirc (AAM)
1	5.5	1	(0.87)	8.5	1	(0.67)	17.5	161	(0.52)
2	5.5	1	(0.87)	7.0	1	(0.67)	19.5	162	(0.52)
3	20.0	162	(0.52)	20.0	162	(0.52)	19.5	161	(0.52)
12	20.0	162	(0.52)	19.5	161	(0.52)	19.0	161	(0.52)

Table 6.6: Random distinguishers for Salsa20 ciphertext.

# of rounds	IV and key reinitialization: once for run — Dieharder (x/20)	once for run — Sts Nist (x/162)	once for run — EACirc (AAM)	for each test set — Dieharder (x/20)	for each test set — Sts Nist (x/162)	for each test set — EACirc (AAM)	for each test vector — Dieharder (x/20)	for each test vector — Sts Nist (x/162)	for each test vector — EACirc (AAM)
1–8	0.0*	0	$n = 104$	0.0*	0	$n = 101$	0.0*	0	$n = 104$
9	1.0	1	$n = 234$	1.5	1	$n = 491$	2.0	1	$n = 121$
10	2.0	13	$n = 188$	3.0	13	$n = 218$	3.0	12	$n = 158$
11	10.0	157	(0.52)	11.5	157	(0.52)	14.0	159	(0.52)
12	16.0	162	(0.52)	17.0	161	(0.52)	17.5	162	(0.52)
13	20.0	162	(0.52)	20.0	162	(0.52)	19.0	162	(0.52)
32	20.0	161	(0.52)	20.0	162	(0.52)	20.0	161	(0.52)

Table 6.7: Random distinguishers for TSC-4 ciphertext.

7 Analysis of Salsa20 output stream

- learns current vectors quicker than other ciphers
- the case of six

8 Distinguishing hash outputs from random stream

- introduction, idea
- hash function candidates from SHA-3

8.1 Hash functions used

- except for 2 (?? source code size, compilation)
- from last phase
- those that could be limited in rounds are tested in weaker variant as well
- differences from Ondrej Dubovec Bc thesis

8.2 Generating binary stream from hash functions

- length set to 256b
- hashing 4 byte counters starting from random value (in fact, cutting each hash in half)

8.3 Determining optimal set change frequency

- previously,we used change every 100 generations
- 100 was taken from Matej Pristak's thesis
- Ondrej proposes 10 as best, however, data is not provided
- interpretation of results (Table 8.1):
    - ???

	change frequency for test vector set							
	5	10	20	50	100	200	500	1000
30 000 g.	(0.614)	(0.614)	(0.607)	(0.602)	(0.599)	(0.598)	(0.591)	(0.582)
run-time	70 m.	52 m.	42 m.	37 m.	32 m.	28 m.	23 m.	20 m.
300 sets	(0.567)	(0.583)	(0.585)	(0.589)	(0.599)	(0.608)	(0.617)	(0.618)
run-time	?? m.	?? m.	?? m.	?? m.	32 m.	57 m.	115 m. 220 m.	

Table 8.1: Determining optimal change frequency for test vector set.

8.4 Results interpretation

- ???

	number of rounds							
	0	1	2	3	4	5	6	7
ARIRANG								
Aurora								
Blake								
Blue Midnight Wish								
Cheetah								
CHI								
CRUNCH								
CubeHash								
DCH								
Dynamic SHA								
Dynamic SHA2								

Table 8.2: Random distinguishers for SHA-3 candidate functions.

9 Conclusions and future work

9.1 Conclusions based on experimental data

- summary of what we did
- control distinguishers (random-random, hr-de, audio)
- estream (round limited ciphers)
- analysis of Salsa20
- sha3 (round limited hash functions)
- different approach than statistical batteries -> possibly new things
- dynamically adapting distinguisher - both advantage and disadvantage
- comparable to statistical tests, however smaller inputs
- speed: slow learning (more computational power needed), fast distinguishing
- problem with interpreting results

9.2 Proposed future work

- deep analyses instead of wide
- possibilities of longer input
  - READX
  - memory circuit
- tools for interpreting results
  - histogram of outputs in nodes
- fixing functions in layers