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 (what are reference points? what result are "normal" and "random"?)
- control distinguisher random-random
- no stable generations found
- average average maximum fitness after test vector change: 0.52
- Dieharder: 20/20
- STS NIST: 188/188
- dependence on number of test set and test set change frequency
- distinguishing Croatia from Germany
  - 6 files of 5 MB from each source
  - read from beginning (initial offset = 0) => same data
  - looking for distinguisher for each pair

		test set size					
		200	500	1000	2000	5000	10 000
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.

6 Distinguishing cipher outputs from random stream

- cipher modes (iv+key initialization frequency)
- tables with results
- 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
- conclusions
  - 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			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	$n = 148$	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			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	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			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		0.0*	0		0.0*	0	
2	0.0*	0		0.0*	0		0.0*	0	
3	0.0*	0		0.0*	0		0.0*	0	
4	0.0*	0		0.0*	0		0.0*	0	
5	0.0*	0		0.0*	0		0.0*	0	
6	0.0*	0		0.0*	0		0.0*	0	
7	0.0*	0		0.0*	0		0.0*	0	
8	0.0*	0		0.0*	0		0.0*	0	
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

- hash function settings (hash length)
- test vector generation method (4 byte counters starting from random point)
- looking for best test set change frequency
- tables with results
- conclusions (???)

9 Conclusions and future work

- conclusions
  - 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
  -
- 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