University of Kent

CRYPTACUS Nov 2017 /
Quam Bene Non Quantum

Statistical analysis of IDQ's Quantis Quantum Random Number Generator

Darren Hurley-Smith & Julio Hernandez-Castro

Introduction

- Darren Hurley-Smith
  - Research Associate
  - School of Computing, University of Kent
- Julio Hernandez-Castro
  - Professor
  - School of Computing, University of Kent
- Current research related to this presentation:
  - Analysing the properties of random number generators
  - Evaluating the effectiveness of certification schemes
  - Identifying independent and robust tests of randomness

Overview of this Research

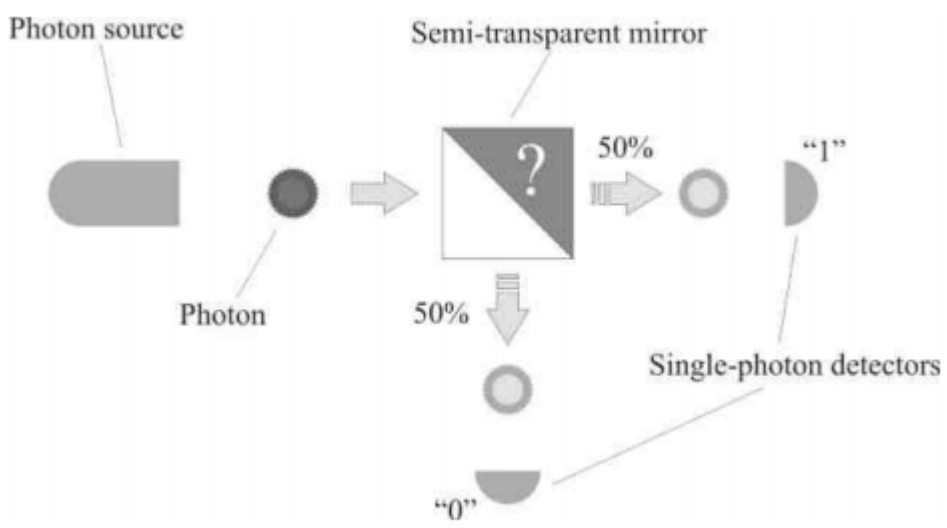
- Previously we've focused on TRNG
  - DESFire EV1 and EV2
  - ChaosKey, Araneus II, TRNG9815, and others…
- We've found flawed TRNG implementations
  - DESFire EV1
  - John Walker's HotBits online service
- Optical Quantum randomness
  - Offers a tamper-proof phenomenon as a source of entropy
  - High-cost, high-speed and claims of high-quality
- What are the attributes of QRNG
  - Does it exhibit any entropy-source biases?
  - What best practices are involved in entropy collection?

IDQ Quantis QRNG Modules

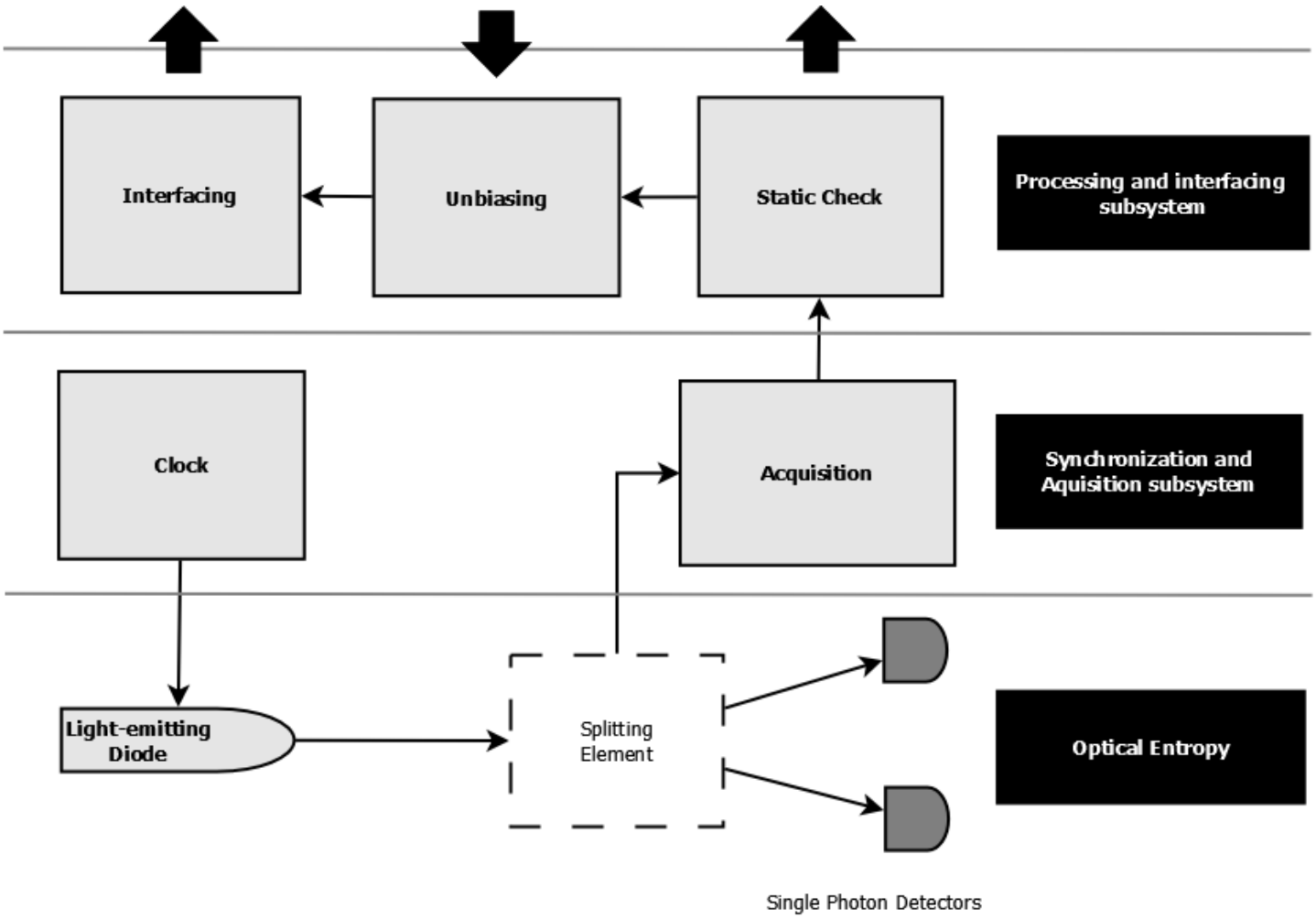
- Three modules tested:
  - 16M (PCI-E 16Mb/s output speed - top)
    - 2990 euros
  - 4M (PCI-E 4Mb/s output speed - middle)
    - 1299 euros
  - USB (4Mb/s output speed – bottom)
    - 990 euros
- Certification provided:
  - Self-certification (Dieharder, NIST SP800-22)
  - Compliance Testing Lab certified (UK)
  - METAS certified
- Real world use-case:
  - Swiss Loterie Romande

Optical Quantum Random Number Generation

- The entropy source pictured below is common to all Quantis devices
  - Image sourced from IDQ Whitepaper on QRNG
  - The 4M and USB possess a single entropy source
  - The 16M mixes the output of 4 entropy sources

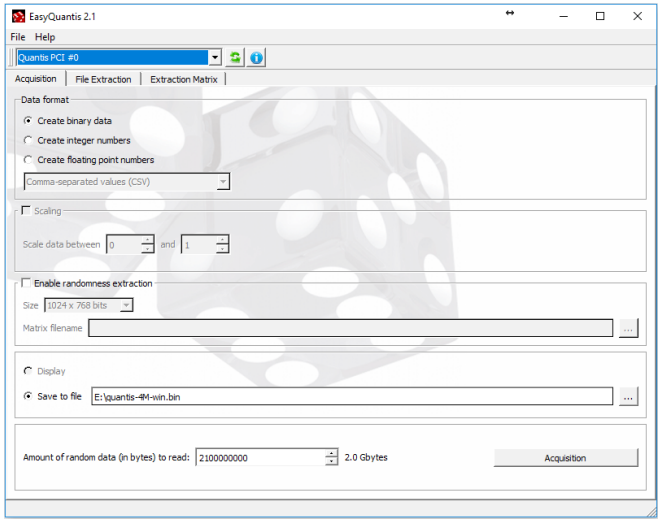


Quantis QRNG Block Diagram



Experiment Set-up

- Initial Data-collection:
  - 3 x 2GiB files collected from each device
  - EasyQuantis GUI Application used to collect data
  - No post-processing



- A more thorough follow-up:
  - 100 x 2GiB collected from each device
  - EasyQuantis command line utility used for collection
  - Raw and post-processed data

Experiment Set-up 2

Results

- Speedtest
  - 16M (15.87Mb/s), 4M (3.86Mb/s), USB (3.96Mb/s)
  - ChaosKey TRNG (3.8Mb/s)
- Dieharder
  - Initial tests show some issues
  - Larger sets show this to be a statistical error
- NIST SP800-22 (STS2.1.2)
  - Initial tests show some issues
  - Larger sets show this to be a statistical error
- ENT
  - Significant byte-level biases found (raw data)
  - Serial correlation of bits shows extremely poor results (raw data)
- TESTUo1
  - Alphabits reports multiple failures (raw data)
  - The Rabbit Battery reports multiple failures (raw data)

Detailed Results (Dieharder/NIST/TestUo1)

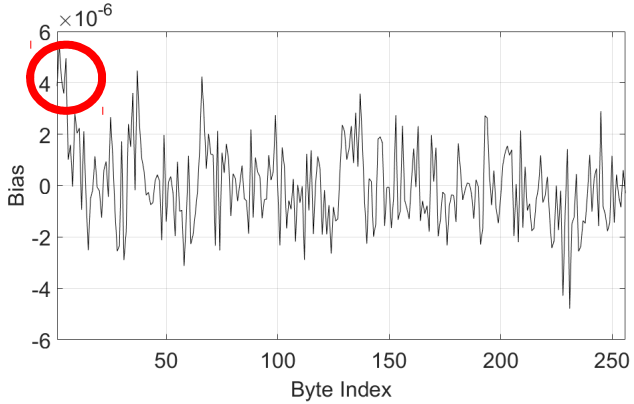
Device	Size	Dieharder	NIST STS 2.1.2	Alphabits	Rabbit
	(GiB)	(Failed/Weak/Passed)	(Passed/Total)	(Passed/Total)	(Passed/Total)
Quantis 16M	2	8 / 11 / 95	182 / 186	7 / 17	26 / 40
	2	6 / 13 / 95	181 / 186	9 / 17	32 / 40
	2	7 / 11 / 96	182 / 186	7 / 17	29 / 40
Quantis 4M	2	0 / 3 / 111	185 / 186	7 / 17	28 / 40
	2	0 / 5 / 109	186 / 186	7 / 17	28 / 40
	2	0 / 6 / 108	186 / 186	7 / 17	27 / 40
	16	0 / 4 / 110	N/A	5 / 17	25 / 40
	32	0 / 3 / 111	N/A	5 / 17	25 / 40
	200	0 / 2 / 112	N/A	5 / 17	24 / 40
Quantis USB	2	0 / 6 / 108	184 / 186	14 / 17	33 / 40
	2	0 / 7 / 107	186 / 186	11 / 17	29 / 40
	2	1 / 6 / 107	184 / 186	10 / 17	30 / 40
ChaosKey	2	0 / 3 / 111	184 / 186	17 / 17	40 / 40
urandom	2	0 / 3 / 111	186 / 186	17 / 17	40 / 40

- Dieharder and NIST are passed
  - 16M is an exception, but further testing suggests these three initial results are anomalous
- Alphabits and Rabbit fail consistently
  - Devices fail slightly different tests more frequently than others
  - ChaosKey (TRNG USB module) passes all tests providing a TRNG baseline
  - Urandom also passes all tests providing a PRNG baseline

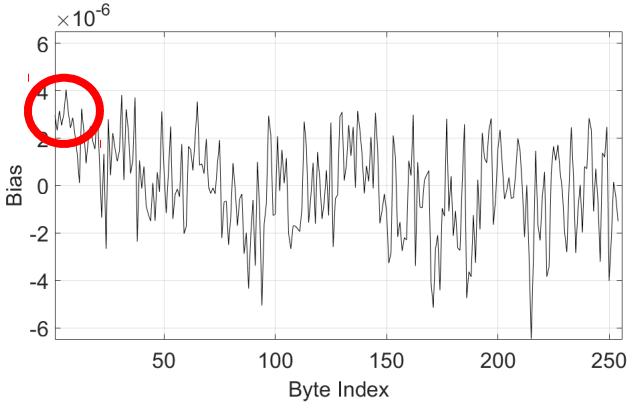
TestUo1 Results in Detail

- Alphabits
  - 4M fails: MultinomialBitsOver, HammingIndep, RandomWalk
  - 16M fails: MultinomialBitsOver, RansomWalk
  - USB fails: MultinomialBitsOver, HammingIndep, RandomWalk
- Rabbit
  - 4M fails: HammingWeight, AutoCor, Run of Bits, RandomWalk
    - 4M fails almost every permutation of the RandomWalk test
  - 16M fails: HammingWeight, AutoCor, Run of Bits, RandomWalk
    - 16M fails fewer permutations of each test
  - USB fails: Fourier3, HammingWeight, HammingIndep, AutoCor, Run of Bits, RandomWalk
    - Unlike Alphabits, the Rabbit battery shows some differences between Usb and 4M

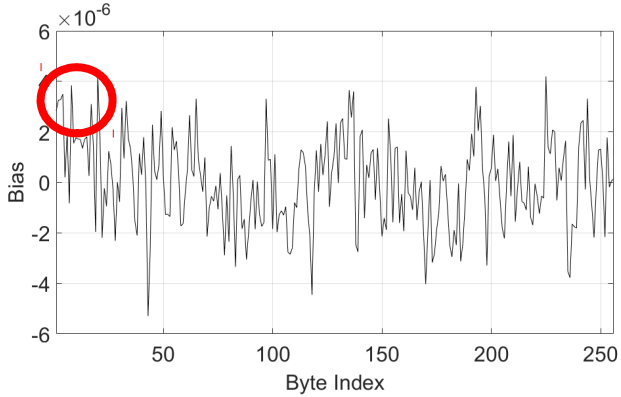
ENT Results in Detail



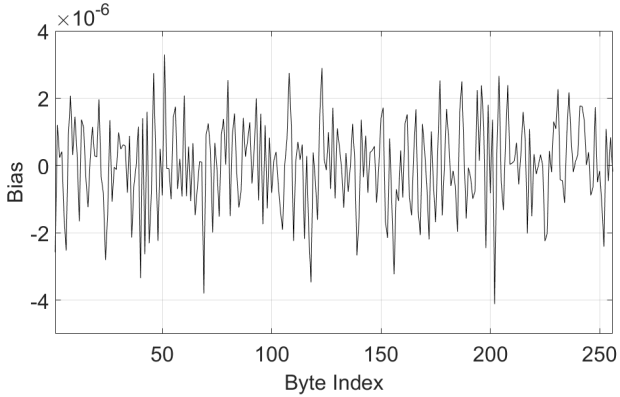
Quantis 16M Bias



Quantis 4M Bias



Quantis USB Bias



URANDOM Bias

ENT Results in Detail 2

File	Size (Bytes)	X2	Serial Correlation	HTCC	p-value
4M 1	2097152000	553.7709	1.40886E-05	0.645181505	0.740305
4M 2	2097152000	489.7084	-3.87639E-05	1.775179538	0.961469
4M 3	2097152000	514.2832	-1.63582E-05	0.749119473	0.772762
4M 4	2097152000	510.2955	5.11633E-05	2.343007275	0.990050
4M 5	2097152000	487.6929	-2.57854E-05	1.180832882	0.880615
4M 6	2097152000	506.4829	1.62732E-05	0.745227468	0.771589
4M 7	2097152000	440.1236	4.11375E-05	1.883879676	0.969640
4M 8	2097152000	548.0064	3.53198E-05	1.617459493	0.946492
4M 9	2097152000	460.4062	-1.63436E-05	0.748449739	0.772560
4M 10	2097152000	540.5753	3.50277E-05	1.60408015	0.945032

4M Byte-level Ent

4M Bit-level Ent

File	Size (Bits)	X2	Serial Correlation	HTCC	p-value
4M 1	16777216000	141.8065	9.36056E-05	12.13267346	0.973823
4M 2	16777216000	103.5581	8.43226E-05	10.9294641	0.970956
4M 3	16777216000	140.0126	8.73252E-05	11.31864453	0.971950
4M 4	16777216000	78.39836	9.30651E-05	12.0544358	0.973654
4M 5	16777216000	38.90776	0.000100731	13.04736851	0.975651
4M 6	16777216000	36.76245	0.000106742	13.82597243	0.977017
4M 7	16777216000	40.74031	9.87896E-05	12.79591623	0.975174
4M 8	16777216000	81.78343	9.64242E-05	12.48953167	0.974568
4M 9	16777216000	44.9761	9.35218E-05	12.11358506	0.973782
4M 10	16777216000	53.16695	9.94391E-05	12.88003447	0.975336

ENT Results in Detail 3

File	Size (Bytes)	X2	Serial Correlation	HTCC	p-value
16M 1	2100000000	373.1276	-0.000023935	1.096839492	0.86313
16M 2	2100000000	302.5997	8.51929E-06	0.390403038	0.65172
16M 3	2100000000	354.927	5.84249E-05	2.677367405	0.99605
16M 4	2100000000	344.8863	-8.28861E-06	0.379831991	0.64781
16M 5	2100000000	350.1756	-2.53896E-05	1.163497738	0.87714
16M 6	2100000000	300.5472	2.75226E-05	1.261242329	0.89581
16M 7	2100000000	333.1116	1.03395E-05	0.473813886	0.68198
16M 8	2100000000	349.3203	1.84284E-05	0.844493199	0.80041
16M 9	2100000000	307.7769	-5.83994E-06	0.267619474	0.6054
16M 10	2100000000	323.3886	4.58495E-06	0.210108902	0.58312

16M Byte-level Ent

16M Bit-level Ent

File	Size (Bits)	X2	Serial Correlation	HTCC	p-value
16M 1	1.68E+10	107.7214	1.25836E-05	1.63101943	0.824928
16M 2	1.68E+10	71.43077	1.16817E-05	1.51412145	0.814207
16M 3	1.68E+10	99.39572	1.34841E-05	1.747736996	0.834573
16M 4	1.68E+10	71.63044	2.15216E-06	0.278952448	0.586592
16M 5	1.68E+10	67.33272	1.83505E-05	2.378498716	0.873314
16M 6	1.68E+10	89.23194	-8.73722E-06	1.132472657	0.769749
16M 7	1.68E+10	60.23114	-1.19597E-06	0.155014929	0.548953
16M 8	1.68E+10	68.17352	7.39666E-06	0.958716239	0.743292
16M 9	1.68E+10	47.17763	7.42624E-06	0.962550644	0.743927
16M 10	1.68E+10	62.60946	2.08663E-05	2.704578136	0.887269

ENT Results in Detail 4

File	Size (Bytes)	X2	Serial Correlation	HTCC	p-value
USB 1	2097152000	436.9514	3.19571E-05	1.464460426	0.927850
USB 2	2097152000	414.8936	1.58698E-05	0.72724485	0.766128
USB 3	2097152000	480.2678	-5.93089E-06	0.271787582	0.606997
USB 4	2097152000	476.2802	-4.53518E-06	0.207828094	0.582235
USB 5	2097152000	431.4787	-4.63862E-05	2.125682795	0.982754
USB 6	2097152000	489.5561	-1.59544E-05	0.731120809	0.767311
USB 7	2097152000	519.5624	1.54902E-05	0.709851536	0.760777
USB 8	2097152000	455.8573	3.17628E-05	1.455555416	0.926627
USB 9	2097152000	440.9796	-1.37956E-05	0.632191614	0.736085
USB 10	2097152000	443.5124	6.59489E-06	0.302215889	0.618633

USB Byte-level Ent

USB Bit-level Ent

File	Size (Bits)	X2	Serial Correlation	HTCC	p-value
USB 1	16777216000	46.03617	8.88723E-05	11.51916149	0.972436
USB 2	16777216000	20.5772	8.75407E-05	11.346569	0.972018
USB 3	16777216000	48.20293	9.39057E-05	12.17157021	0.973906
USB 4	16777216000	37.68492	9.72978E-05	12.61123074	0.974812
USB 5	16777216000	42.02821	8.59015E-05	11.13411296	0.971487
USB 6	16777216000	40.94392	9.69857E-05	12.57077812	0.974731
USB 7	16777216000	49.70702	9.89094E-05	12.82012646	0.975221
USB 8	16777216000	57.68654	8.80525E-05	11.41291091	0.972180
USB 9	16777216000	48.25115	8.28145E-05	10.73398723	0.970430
USB 10	16777216000	20.40718	9.91219E-05	12.84766418	0.975274

Discussion

- Quantis raw output is biased
  - IDQ respond stating that post-processing is required
  - Post-processing is listed as optional in their product manual
  - Robust randomness at 16Mb/s is claimed:
    - Post-processing cuts this by 50-65% depending on CPU model
- Post-processing solves all identified problems
  - Post-processing is effective
  - All tests are passed by post-processed data, for all devices
  - Software implementation means that trust must be placed in both device and independent software
- Suitability for IoT is in question
  - Post-processing is CPU and memory intensive
- Yet more evidence of lax certification
  - Diehard over 10x1MB of data garnered official approval
  - Dieharder and NIST indicate serious problems, but miss flaws
  - Better, independent tests must be identified

Conclusion

- Quantum random number generation
  - Inherent bias is optical QRNG is a known phenomenon
  - Many devices claim random output despite this
  - Randomness is achievable, but requires supporting hardware/software
- Post-processing should be accounted for
  - One shouldn't claim robust randomness at speeds prior to post-processing
  - Post-processing is NOT optional
  - Users must be aware that their potential attack surface increases
    - Manipulation of the input matrix can affect output predictably
    - Manipulation of the post-processing algorithm can insert predictable values whilst preserving 'randomness'
    - Software implementations mean that attackers can target the PC
      - This is must easier than trying to attack the sealed device!
- Future work
  - Analysis of the Comscire QRNG
  - Analysis of post-proc matrix manipulation to stealthily insert known values into an output stream

Thank you for listening.

Questions?

THE UK'S EUROPEAN UNIVERSITY

www.kent.ac.uk

University of Kent