# Chapter 10 Challenges in Certifying Small-Scale (IoT) Hardware Random Number Generators

**Darren Hurley-Smith and Julio Hernandez-Castro**

**Abstract** This chapter focuses on the testing and certification of Random Number Generators (RNG). Statistical testing is required to identify whether sequences produced by RNG demonstrate non-random characteristics. These can include structures within their output, repetition of sequences, and any other form of predictability. Certification of computer security systems draws on such evaluations to determine whether a given RNG implementation contributes to a secure, robust security system. Recently, small-scale hardware RNGs have been targeted at IoT devices, especially those requiring security. This, however, introduces new technical challenges; low computational resources for post-processing and evaluation of on-board RNGs being just two examples. Can we rely on the current suite of statistical tests? What other challenges are encountered when evaluating RNG?

## 10.1 Introduction

Randomly generated values are sought after for a variety of applications, in which they are often vital. Cryptographic systems require random values to ensure that generated keys are unpredictable, making brute force attacks against those keys unfeasible. Even in the entertainment industry, there is a demand for randomness: lotteries and games both rely on random number generation to guarantee the fairness of the game in question.

However, random number generation is a non-trivial task. Deterministic Random Number Generators (DRNG), also known as Pseudo-Random Number Generators (PRNG), are incapable of truly random output [514]. PRNG achieve an *appropriate degree of randomness* by using an initial seed value to populate a proportionally far longer sequence of apparently random output. This form of random number generation is only unpredictable if the seed value remains unknown. To this end,

D. Hurley-Smith (✉) · J. Hernandez-Castro
University of Kent, Canterbury, UK
e-mail: darren.hurley-smith@rhul.ac.uk

G. Avoine, J. Hernandez-Castro (eds.), *Security of Ubiquitous Computing Systems*,
https://doi.org/10.1007/978-3-030-10591-4_10

most PRNG algorithms are periodically re-seeded from a natural source of entropy. The primary benefit of PRNGs is that they are usually extremely fast, especially when compared to the natural entropy sources used to seed them. This makes them highly attractive for use in computer systems, and in applications requiring high-volume RNG.

Physical sources of entropy can provide what is referred to as *true randomness*. True Random Number Generators (TRNG) use a broad array of different entropy sources as their key component but share several common characteristics. They do not require seeding to generate randomness and use a natural phenomenon as their entropy source. TRNGs can be classified further, as classical or Quantum Random Number Generators (QRNGs). To simplify matters, TRNG will refer to classical methods, and QRNG will refer to quantum methods from this point. TRNG utilize microscopic phenomena that generate statistically random noise signals. The photoelectric effect and thermal noise are two examples of classical entropy sources. QRNG operate on similar principles but instead make use of quantum phenomena. These include photon-counting, using a beam-splitter, or the observation of quantum shot-noise in MOS/CMOS devices.

All random number generators can be evaluated using statistical test batteries. Dieharder, Federal Information Processing Standard (FIPS) 140-1/2, and National Institute of Standards and Technology (NIST) SP800-22 [448] represent the three most common test batteries used for professional testing of random number generators. Manufacturers often use such tests to demonstrate the correct functioning of their products, but they are also used by third-parties to independently verify the randomness of a device. NIST and Common Criteria [407] provide guidelines and tests that have been independently developed to ascertain whether an RNG is non-random. These tests evaluate RNGs by identifying whether there is any observable bias, structure or predictability in an RNG's output. It is not possible to identify randomness, but non-randomness can be detected. Certification schemes make use of such tests to publicly acknowledge the robustness of computational security systems. Specific methodologies have been devised to guide and ensure the quality of these evaluations in the area of RNG validation.

Significant trust is placed in statistical testing to determine whether an RNG provides sufficiently random output. The aim of this chapter is to demonstrate that the challenges of statistical testing of randomness are far from solved. We evaluate a selection of contemporary TRNG to highlight issues in data collection, test correlation and the overuse of older test batteries to the exclusion of newer tests. As minuscule, integrated TRNG become more prolific through their use in Internet of Things (IoT) products, these considerations become all the more important.

The following sections discuss, in order: certification of RNGs and the standards/testing procedures that apply, the challenges faced during the collection of data from RNGs, and two sets of experimental results demonstrating issues in the appropriate selection of statistical tests for RNG evaluation.

## 10.2 Certification, Standards, and Testing

Many companies employ their own testing teams, to whom the responsibility of carrying out company mandated quality control falls. ID Quantique (IDQ) and NXP are two examples, both of whom perform varying degrees of testing on their products. In the case of products implementing cryptography, RNG testing is vital for the validation of the cryptosystem in question. However, in-house testing is insufficient for certification, with the exception of self-certification (as performed by IDQ). Testing must be performed by a third-party to ensure impartiality.

NIST is one example of a standards and testing institution. This US institute concerns itself with the advancement of measurement science, standards and technology. This body does not conduct testing or reward certificates itself but is responsible for the publication and impartial development of statistical test suites for randomness tests. Special Publications (SP) are created to circulate accepted developments in the field of RNG testing and formal verification of RNG. Of particular note are SP800-90B [449] and SP800-22 [448]. SP800-90B details specific tests for the entropy source and final outputs of PRNG and TRNG. SP800-22 details an extensive test battery suitable for use over PRNG and TRNG (including QRNG by association with TRNG).

Common Criteria (CC) is an international standard (ISO/IEC 15408). Unlike the NIST SP documents discussed previously, CC is a broad framework for the verification of computer security systems [407]. Functionality, construction, and assurance requirements are the core tenets of the CC. It is important to emphasize that this is a whole-system-security verification: RNG testing is only part of a larger verification process. However, it can be argued that RNG validation is a keystone for the certification of a computer-based security system. If the RNG is incapable of providing the appropriate output, then it is unlikely that the security system will be robust to the degree demanded by the CC.

To differentiate between different applications and their security requirements, the CC has developed the Evaluation Assurance Level (EAL) scheme. These numbered levels, from 1 to 7, reflect an increasing security requirement. At level 1, testing is cursory and reports provided by manufacturers are acceptable. As higher certifications are sought, more third party and design-stage tests by third parties are required. At levels 5+, spot checks of manufacturing plants and implementation of security critical systems are performed. NXP produces two CC EAL certified devices: the DESFire EV1 (EAL4+), and the DESFire EV2 (EAL5+).

The test methodology employed by the CC when testing RNGs is outlined in AIS-31 [327]. AIS-31 outlines the test methodology for entropy sources in computer-based security systems [327]. AIS-20 is referred to as the source of information for recommended tests and parameters for TRNG evaluation. Both documents have a TRNG focus, as they are aimed at the evaluation of the formal verification of entropy sources, not the PRNG algorithms that they may seed. As a result, hardware RNGs are the focus of these documents.

**Table 10.1** Standards applied in the testing of selected RNGs

| Manufacturer | Device | Cost € | Entropy source | Certifications/tests |
|---|---|---|---|---|
| NXP | DESFire EV1 | 0.59 | Not disclosed | CC EAL4+ |
| NXP | DESFire EV2 | 1.25 | Not disclosed | CC EAL5+ |
| IDQ | Quantis 16M | 2900 | Beam splitter | NIST SP800-22, METAS, CTL |
| IDQ | Quantis 4M | 1299 | Beam splitter | NIST SP800-22, METAS, CTL |
| IDQ | Quantis USB 4M | 990 | Beam splitter | NIST SP800-22, METAS, CTL |
| Comscire | PQ32MU | 1211 | Shot noise | NIST SP800-90B/C |
| | | | | NIST SP800-22 |
| | | | | Diehard |
| Altus Metrum | ChaosKey | 45 | RBSJ[a] | FIPS 140-2 |

[a]Reverse biased semiconductor junction

Table 10.1 shows a selection of RNGs and their associated certifications. CC EAL, METAS, CTL, and FIPS 140-2 are applicable as certifications from their respective institutions. NIST SP800-22 indicates that the NIST methodology and test battery were applied when testing the RNG in question (whether internally or externally). Any RNG testing process requires a set of statistical tests. One of the earliest examples of a statistical test battery for randomness is Marsaglia's Diehard battery [393]. NIST SP800-22 provides a more expansive series of tests developed by Rukhin et al. [508]. The NIST battery contains 15 tests, which are evaluated in terms of uniformity and proportion of *p*-values for each test. There has been some criticism of the accuracy of these results. Marton and Suciu observed that false alarms were common and that more tests that SP800-22 suggests can be failed by otherwise robust RNGs [396]. NIST itself states that any failure is cause for further investigation, but does not suggest any specific follow up procedures for RNG testing. It is implied that further data collection and testing a larger number of target devices are initial approaches to the problem.

Dieharder is an extension of Diehard, integrating the SP800-22 tests and the original Diehard tests [116]. This brings the battery up to a total of 30 tests, with 76 variant tests in total. This battery requires a much larger body of test data than its predecessors. To test a stream of data with no rewinds with every test in the suite, one must collect 228 GB of data. This is far beyond the recommended parameters suggested by NIST and CC. A 4 GB sample would rewind 57 times under the same test conditions. If a sequence of repeats during a single execution of a given test, type-1 errors may be introduced. The test may report such repetition as a violation of its definition of randomness, and identify the sequence as non-random, when in fact it was just insufficiently large. This highlights the importance of appropriate data collection.

TestU01, developed by L'Ecuyer and Simard, is more of an RNG developer's toolbox than test battery [362]. However, it incorporates 5 different test batteries: Alphabits, Rabbit, Small Crush, Crush, and Big Crush. Each battery has a differing number of tests. Alphabits and Rabbit operate over bits, whilst the Crush batteries operate over floating point numbers between 0 and 1. Alphabits, Rabbit, and Small Crush complete in minutes over samples of 2 GB in size. Big Crush requires a large amount of data (or a constant stream of input from the target device) and can take hours to complete. McCullough et al. identify some potential issues with this toolset. Some tests are only able to read 32 bits and are more sensitive to errors in their most significant bits than their least significant bits [403]. To resolve this, they suggest that tests are performed over the sequences forwards and backwards. The issue here is that a test on live data cannot be performed in this manner. This limits many tests and prevents them from being used as live tests.

Another class of tests exists; continuous tests. These tests are designed to identify whether there have been hardware failures that lead to corruption or cessation of the entropy stream. FIPS 140-1/2 are designed with hardware in mind [121]. Both tests suites can be implemented in the circuitry of an RNG, providing a constant series of results regarding the health and functionality of the device. A core requirement of any continuous test is that no RNG should output two identical *N* bit blocks in succession. If this condition is not met, the device should cease function immediately and alert the user that it is not performing as expected. However, this does not detect more subtle flaws. The astute reader may also have deduced that requiring that no two *N* bit blocks be identical actually results in reduced entropy. This has an impact on the legitimacy of such tests when considering that the definition of an ideal RNG is one that is completely unpredictable. These tests are likely to be implemented alongside IoT TRNG implementations due to their efficient implementation in hardware, carrying the previous concerns to millions of potential devices.

The usage of NIST, Dieharder, TestU01, and other statistical test batteries can vary between institutions. NIST SP800-22 outlines minimum sample sizes and Dieharder implies these by rewinding samples if insufficient data is provided. However, during self-certification, some companies have been found to test small samples, below the suggested guidelines. This can cast doubt over the validity of their findings.

## 10.3 Challenges in Data Collection

For standalone RNG, data collection may be simple. However, there are no official certifications for standalone RNG. FIPS and CC both certify whole security systems, not individual elements, so even though RNG testing is key to this process, a standalone RNG that passes these tests still cannot be certified. Regardless, RNG testing as a part of whole system certification is a critical consideration. Data collection from certain integrated RNGs may not be trivial. As IoT devices represent a whole-system security implementation, they may be certified; RNG evaluation forms a critical part of any such evaluation process.

Black-box design is often employed by companies using licensed technology, or who need to protect their Intellectual Property (IP). This means that schematics of their security implementation, including RNG, may not be publicly accessible. For lower EAL awards, such obfuscation of technical detail may extend to inspectors and CC officials. At higher levels, non-disclosure agreements are required as a part of the certification contract between the petitioning company and the evaluating body. Such arrangements are expensive. Inspectors and independent testers have to be compensated for their times and the cost falls to the company requesting an evaluation at a given EAL. As a result, self-certification is common.

The speed with which an RNG may be read depends on a great many factors. In situations where the RNG is fully integrated, there may be additional overheads such as post-processing, use of a PRNG to *clean* TRNG output used as an entropy source, or simply a hard limit on output size and speed. A poignant example of this is the DESFire EV1 and EV2. These RFID cards do not directly expose their internal TRNG to the user, requiring that the user extracts random numbers using the authentication protocol instead. This protocol requires that both the card and reader exchange random values as part of their authentication handshake [289]. The 16 bytes values transmitted by the card can be collected and stored in a file for analysis using statistical tests for randomness [288]. This is a time-consuming procedure, as Table 10.2 shows. To collect 64 MB of data from the DESFire cards, approximately 12 days were required. The primary bottleneck in this process was the need to complete the authentication protocol before a second handshake could be initiated to gather additional 16-byte sequences. Attempting to terminate the protocol by switching off the reader, thus resetting the card, proved to be even more time-intensive [288]. This issue is shared by IoT devices, many of which implement integrated TRNGs.

IoT devices have a plethora of ways in which PRNG and TRNG may be implemented. The FRDM K64F board implements a TRNG, though the output is limited to making calls internally for use, or outputting values over an I/O pin in the form of unsigned integers. Though significantly faster than the EV1 and EV2, this is still much slower than most standalone TRNGs. The Red Bear Duo does not implement a local entropy source. An on-board PRNG must be supplied with off-device entropy, with no checks or continuous tests performed on-device. In a full-system implementation, such a device can make it difficult to identify where the flaw in its RNG occurs.

**Table 10.2** RNG output speed of selected devices

| | Sample size (MB) | Mean data rate (bit/s) |
|---|---|---|
| DESFire EV1 | 64 | $4.93 \cdot 10^2$ |
| DESFire EV2 | 64 | $4.90 \cdot 10^2$ |
| Quantis 16M | 2100 | $1.27 \cdot 10^8$ |
| Quantis 4M | 2100 | $3.08 \cdot 10^7$ |
| Quantis USB 4M | 2100 | $3.11 \cdot 10^7$ |
| Comscire PQ32MU | 2100 | $2.48 \cdot 10^8$ |
| ChaosKey | 2100 | $3.07 \cdot 10^7$ |

The standalone generators (Quantis, Comscire and ChaosKey entries in Table 10.2) are substantially faster, making data collection trivial by comparison. However, this does not mean that samples of appropriate size were tested.

SP800-90B states that an entropy source must provide 1,000,000 bits of sequential output for testing [449]. Concatenation of smaller sequences is tolerable if contiguous output to that size is not possible, but is undesirable. 1000 such sequences must be concatenated, according to NIST guidelines. SP800-22 extends these requirements by recommending that 100 samples of the aforementioned size are tested to validate the results [508]. AIS-31 and AIS-20 do not stipulate minimum sample sizes. John Walker states that, in their default configuration, the Diehard tests should be run over at least 100 MB of data [568].

With this in mind, the test reports of several TRNG manufacturers can be more thoroughly analyzed. IDQ states that their Quantis devices pass the Diehard and NISTSP800-22 batteries with no failure.[1] SP800-22 tests were conducted over 1000 samples of 1,000,000 bits in length. A significance level of 1% was maintained throughout this process. Diehard was used over a single sample of $1 \cdot 10^9$ bits. Our own tests confirm that IDQ's report of no failures is true, even for larger samples (ours were 2.1 GB in size). In this case, IDQ is a good example of a test protocol that is in line with the recommendations of test developers.

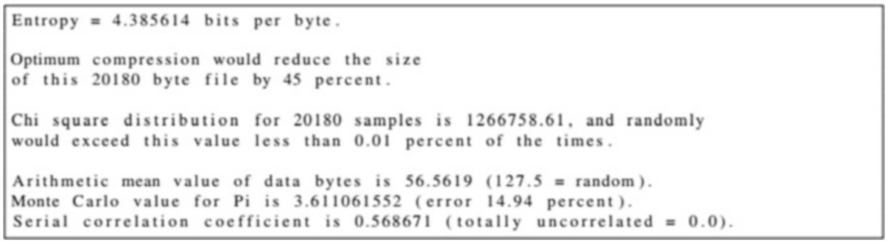
Comscire's PQ32MU, a QRNG that uses shot-noise as an entropy source, is a different story. Their NIST-Diehard report[2] shows that the number of tests has been reduced. The reduced sample size is one issue, but reducing the number of tests can result in the loss of certain capabilities. Unless the removed tests are wholly redundant, it is likely that their removal will impact the capability of the battery to detect certain types of non-randomness. The insufficient sample size is cited as the reason for excluding those tests. Comscire only tests this QRNG using 2 samples; one of $8 \cdot 10^7$ bits and another of $1 \cdot 10^6$ bits. This is drastically below the suggested sample size for Diehard. Even though these samples meet the requirements of NIST SP800-90B in the most basic sense, they still fall short of SP800-22's additional recommendations requiring the testing of at least 100 samples. Considering the ease with which samples can be generated from standalone RNGs such as these, it is surprising that a more robust test process is not used.

## 10.4 Appropriate Selection of Tests

The correlation between tests in a battery, and as a whole if the evaluation methodology involves multiple test batteries, must be considered. Statistical tests have a limited range of issues that they are able to identify in the target RNG. Test

[1]https://marketing.idquantique.com/acton/attachment/11868/f-004c/1/-/-/-/-/Randomness%20Test%20Report.pdf.

[2]https://comscire.com/files/cert/comscire-pq32mu-nist_diehard-validation-tests.pdf.



```
Entropy = 4.385614 bits per byte.

Optimum compression would reduce the size
of this 20180 byte file by 45 percent.

Chi square distribution for 20180 samples is 1266758.61, and randomly
would exceed this value less than 0.01 percent of the times.

Arithmetic mean value of data bytes is 56.5619 (127.5 = random).
Monte Carlo value for Pi is 3.611061552 (error 14.94 percent).
Serial correlation coefficient is 0.568671 (totally uncorrelated = 0.0).
```

**Fig. 10.1** Example of Ent default output in byte mode [272]

batteries are intended to mitigate this issue by providing many statistical tests that evaluate different aspects of the target RNG, providing a broader analysis.

Hernandez-Castro et al. identify a degree of correlation between tests in the Ent battery. The Ent battery is a simple set of tests included in most Linux distributions as a simple statistical testing tool [568]. Ent includes tests for estimated entropy, compression, $\chi^2$, arithmetic mean, Monte Carlo $\pi$, and serial correlation. Bit and byte level tests can be run over target sequences. Figure 10.1 shows the output of the Ubuntu 16.04 Ent utility in byte mode.

By degenerating an initially random sequence using a genetic algorithm, Hernandez-Castro et al. were able to observe the test results of Ent as the sequence slowly became more ordered and predictable [272]. The results demonstrate that many of the Ent tests have a degree of correlation. Entropy and compression tests analyze the same general attributes, both performing linear transformation and ceiling operations on a sequence. The $\chi^2$ and excess statistics provided by the $\chi^2$ test are also closely correlated. The conclusion of the paper recommends that the excess and compression statistics should be discarded.

Soto et al. explore the degree of correlation between tests in the NIST SP800-22 battery. Their work finds that the range of attributes evaluated by SP800-22 may be insufficient to recognize issues [538]. TRNG and QRNG are particular issues, as many examples of these RNGs have been developed since the development of SP800-22. Soto describes the independence of tests in this battery as an open problem.

Turan et al. provide a more recent analysis of SP800-22. Their work finds that the frequency, overlapping template (input template 111), longest run of ones, random walk height, and maximum order complexity tests produce correlated statistics [560]. This issue is most evident when using small samples or block sizes. Georgescu et al. build on Turan's work, identifying and examining the open problems in SP800-22 test correlation. The sample size is found to have a significant effect on the correlations between tests. The correlations identified by Turan et al. are confirmed, and their relationship with sample size explored in greater depth [226]. Such results demonstrate that every element of an RNG test methodology must be carefully examined to ensure a meaningful and unbiased result. Georgescu et al. conclude by stating that better tests than those implemented in SP800-22 may exist, as that battery is now quite old.

Researchers have commented on the ambiguity of SP800-22's hypothesis and statistical output, stating that more descriptive test output is required. Zhu et al. propose a *Q-value*, computed using test statistics prior to their consolidation to $p$-values [601]. The proposed statistic is more sensitive to total variation distance and Kullback-Leiber divergence. This overcomes some of the issues caused by correlations between the non-$\chi^2$ level 2 tests of SP800-22 [601].

Dieharder implements many of the SP800-22 tests. As a result, it shares many of the criticisms levelled at SP800-22 [206]. TestU01 is a more recent battery aimed at allowing researchers to develop and evaluate their own RNGs, especially TRNGs. There is little critical literature regarding this battery at present, so the independence of tests in TestU01 is an open question at this time. Turan et al. comment on the presence of some tests that they have found to be correlated being implemented in the Crush batteries of TestU01 [560].

The diversity of a test methodology is related to, but separate from, the independence of tests. Where independence is a measure of how related the results from a set of tests may be, diversity is a measure of how many methods of evaluation are used in the analysis of an RNG. A common observation is that the isolated use of $p$-values is insufficient to fully characterize the randomness (or lack thereof) of a sequence. Research by Hurley-Smith et al. explores TRNG and QRNG in detail to identify flaws that were not detected by the most commonly used test batteries. In these analyses, test correlation and diversity are key topics.

### *10.4.1 Randomness Testing Under Data Collection Constraints: Analyzing the DESFire EV1*

The first of these in-depth analyses was conducted over the Mifare DESFire EV1, an RFID card produced by NXP [379]. The DESFire EV1 is used as a part of the Transport for London (TfL) Oyster card scheme, as well as other loyalty and e-wallet schemes throughout Europe. As a device that can store cash value, it requires robust security to foster trust among vendors and users. The EV1 has achieved an EAL4+ certification, based on its full security implementation.

Table 10.3 shows the Dieharder results for 3 DESFire EV1 cards. As mentioned previously, data collection from the EV1 is challenging, requiring 12 days to obtain 64 MB of data. As a result, this was the largest amount of data able to be collected. A total of 100 cards were tested, with all 100 passing. The 3 cards shown in this table show the $p$-values reported by the Dieharder tests for all tests that can be performed on 64 MB of data without rewinds.

Card 3 shows a single failure of the Dieharder battery, for the count the ones test. However, this was not reproduced by any other card that was tested. Therefore, it is reasonable to conclude that the Dieharder battery does not identify any significant degree of non-randomness in the tested sequences.

Table 10.4 shows the pass rates for NIST tests. All SP800-22 tests were used over the EV1 samples we collected.

**Table 10.3** Dieharder results [289]

| Test | Card 1 | Card 2 | Card 3 |
|---|---|---|---|
| Birthday spacings | 0.18194520 | 0.61105583 | 0.78263630 |
| Overlapping permutations | 0.38044164 | 0.58693289 | 0.44201308 |
| 32 × 32 Binary rank | 0.42920693 | 0.23409500 | 0.55699838 |
| 6 × 8 Binary rank | 0.31311490 | 0.32387215 | 0.66137580 |
| Bitstream | 0.97724174 | 0.18743536 | 0.59532464 |
| Count the 1’s (stream) | 0.17108396 | 0.74984724 | 0.87214241 |
| Count the 1’s (byte) | 0.65870385 | 0.01287807 | **0.00020194** |
| Parking lot | 0.18078043 | 0.24200626 | 0.38128677 |
| Minimum distance (2d sphere) | 0.76328000 | 0.95091635 | 0.34980807 |
| 3d sphere (minimum distance) | 0.23871272 | 0.20826216 | 0.39340851 |
| Squeeze | 0.62598919 | 0.08843989 | 0.77057749 |
| Runs | 0.99778832 | 0.62043244 | 0.90550208 |
| | 0.44719093 | 0.91228597 | 0.04870531 |
| Craps | 0.54077256 | 0.92769962 | 0.91803037 |
| | 0.57614807 | 0.94245583 | 0.95209393 |

The bold values indicate those tests which fail, but such a degree that they are well outside the bounds of confidence established by NIST (or in the case of Ent, our extrapolation of the NIST SP800-22 confidence bound of $a = 0.01$)

**Table 10.4** NIST SP800-22 results [289]

| Test | Card 1 | Card 2 | Card 3 |
|---|---|---|---|
| Frequency | 198/200 | 200/200 | 197/200 |
| Block frequency | 196/200 | 199/200 | 194/200 |
| Cumulative sums | 2/2 | 2/2 | 2/2 |
| Longest run | 196/200 | 198/200 | 198/200 |
| Rank | 198/200 | 199/200 | 197/200 |
| FFT | 197/200 | 199/200 | 198/200 |
| Non-overlapping template | **147/148** | 148/148 | 148/148 |
| Overlapping template | 198/200 | 198/200 | 198/200 |
| Universal | 198/200 | 198/200 | 198/200 |
| Approximate entropy | 197/200 | 198/200 | 196/200 |
| Random excursions | 8/8 | 8/8 | 8/8 |
| Random excursions variant | 18/18 | 18/18 | 18/18 |
| Serial | 2/2 | 2/2 | 2/2 |
| Linear complexity | 199/200 | 197/200 | 199/200 |

The bold values indicate those tests which fail, but such a degree that they are well outside the bounds of confidence established by NIST (or in the case of Ent, our extrapolation of the NIST SP800-22 confidence bound of $a = 0.01$)

Cards 1 and 3 both show some borderline results, notably in the Runs and Non-overlapping template tests. However, the majority of cards (98 of 100) passed this battery. This was a cause for concern: any failure is a cause for further investigation as stated by SP800-90B. As a result, further analysis was deemed necessary, and the

**Table 10.5** Mifare DESFire EV1 ENT results for 64 MB of TRNG output [289]

| | Card 1 | Card 2 | Card 3 | Optimal |
|---|---|---|---|---|
| Entropy | 7.999969 | 7.999989 | 7.999972 | 8 |
| Optimal compress. | 0 | 0 | 0 | 0 |
| $\chi^2$ | **2709.10** | **973.07** | **2470.32** | 256 |
| Arith. mean | 127.492921 | 127.500582 | 127.5006 | 127.5 |
| Monte Carlo $\pi$ est. | 3.14167 | 3.142019 | 3.141909 | 3.14159 |
| S. correlation | 0.000008 | 0.000045 | 0.000093 | 0.0 |

The bold values indicate those tests which fail, but such a degree that they are well outside the bounds of confidence established by NIST (or in the case of Ent, our extrapolation of the NIST SP800-22 confidence bound of $a = 0.01$)

humble ENT battery was used as a starting point for a more generalized approach to our EV1 TRNG evaluation.

Considering Hernandez-Castro et al.'s work on the independent of Ent tests, the compression and excess statistics should be discarded. However, the full results of the Ent battery over 3 EV1 cards are shown in Table 10.5 for the sake of completeness.

All tests are passed, with the exception of the $\chi^2$ test. For the 3 64-MB samples shown in Table 10.5, the $\chi^2$ statistic is exceptionally poor. By comparison, a sequence that passes this test should have a $\chi^2$ statistic of between 220 and 305. Even at a sample size of 1 MB, 100 DESFire EV1 cards failed this test. These results show that the values in the tested sequences are not uniformly distributed: there is a bias towards some byte values and away from others. Considering that the $\chi^2$ test is such a trivial (and widely used) test of the distribution of values in a sequence, it is surprising that it would highlight issues in the output of the EV1's TRNG while Dieharder and NIST SP800-22 do not.

Non-uniform distribution of bytes is not an automatic indicator of non-randomness. It is not a good indicator of randomness, but it is also possible for a true source of randomness to produce a slightly biased sequence. However, as per the guidelines of AIS-20 and SP800-90B, a TRNG should provide an output that is functionally equivalent to that of a cryptographic PRNG. As a result, non-uniform byte distribution is a concern. The fact that there is bias is an important observation, but more important is the analysis of that bias.

Figure 10.2 provides a deeper examination of how bias is expressed by the TRNG output of 100 DESFire EV1 cards. Figure 10.2a shows the mean bias of 100 1-MB samples. The extreme deviation from the expected distribution of values is apparent in the square-wave of the plot. The expected distribution should result in a noisy, relatively evenly distributed set of byte values. A bias in the order of $10^{-5}$ is observed, with an almost evenly distributed bias among values that are deviated above or below the normal. To be precise, 127 values are biased above the normal, and 129 are biased below the normal.

Figure 10.2b refines the observations of the previous graph. Fourier approximation of the bias reveals that the distribution of byte values has a period ($w$) of

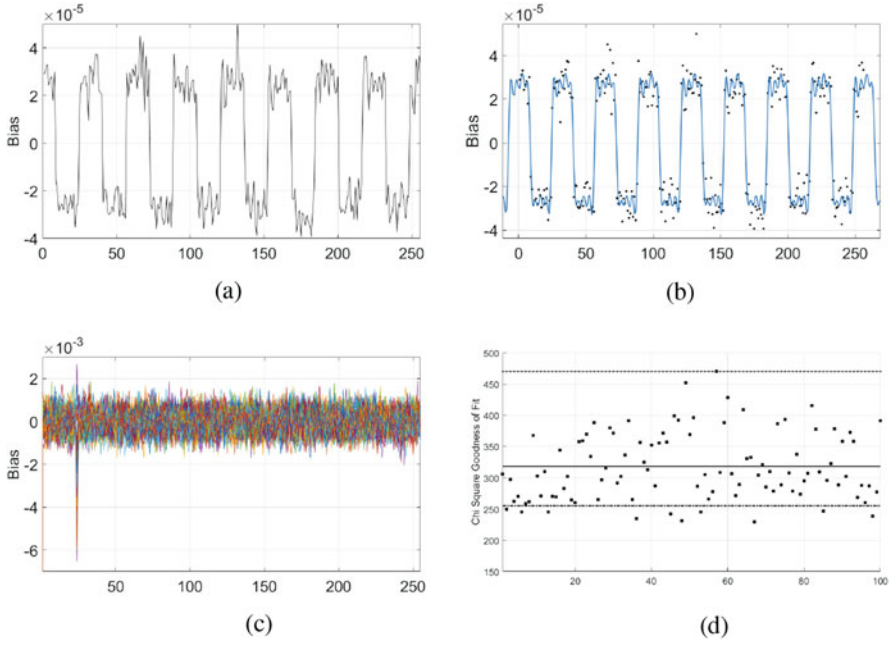


**Fig. 10.2** Analysis of DESFire EV1 bias [289]. (**a**) Mean bias of 100 1-MB samples. (**b**) Mean Fourier approx. of 1 MB samples. (**c**) Mask test results. (**d**) $\chi^2$ scores for 100 1-MB samples

−31.9918. This results in 8 oscillations throughout the 256 possible byte values, with a shift across the normal, observed every 32 values (approximately). Statistical analysis of the possible distribution of bits within these byte values shows that the under-occurrence of a specific bit-sequence can result in this very particular form of bias.

Figure 10.2c provides the results of a so-called *mask test*. The purpose of this test is to XOR each byte of a sequence with an 8-bit sequence, ranging from *0000000* through all intervening values to *11111111*. The sum of all sequences that resolve to zero after the XOR operation records the occurrence of that bit-sequence throughout a sample. This graph shows the composite of 100 1-MB sequences tested in this manner. It is immediately apparent that there's a significant deviation from the normal for mask *00011000*. For all cards, and for both 64 and 1 MB samples, this bias was observed. Following our responsible disclosure to NXP, it was suggested that this bias may be caused by an incorrectly implemented whitening-function: a function usually intended to remove bias from TRNG output.

Figure 10.2d shows the distribution of the $\chi^2$ statistic for 100 1-MB EV1 samples. The statistics are proportionally lower than those seen for the 64 MB samples (Table 10.5). This is because sample size has a direct effect on the expression of bias within a sequence. Early experiments conducted by Hurley-Smith et al. demonstrated that the bias of the DESFire EV1 could not be observed at sample sizes smaller than 7.5 KB [288, 289]. This emphasizes the point made in Sect. 3 regarding data collection. The amount of data collected needs to meet a minimum size to reliably identify issues in the RNGs being tested. This minimum threshold is test-specific, requiring that the highest minimum sample size is identified by analysts prior to conducting an evaluation of an RNG. Furthermore, these experiments showed that even well-respected, proven statistical test batteries such as Dieharder and NIST SP800-22 were unable to identify the issue with the DESFire EV1. It is clear that it is possible to design a TRNG to pass these tests, but is it wise to rely on tests that can be designed for? Does designing to meet the finite and narrow requirements of Dieharder and NIST SP800-22 actually provide any guarantees of randomness? We argue that it does not.

### *10.4.2 Identifying Issues with Quantum Random Number Generators*

The EV1 experiments provided an introduction to issues in using well-established statistical tests to identify non-randomness in TRNG. QRNGs are currently too large for RFID card or IoT device implementations, but miniaturization of quantum entropy sources is proceeding quickly, and proposals for IoT-scale QRNG have already been published. However, there are several open problems with the current generation of QRNGs and their evaluation.

Even when sample collection is not a problem, there can be issues. IDQ's Quantis range of QRNGs is based on an optical quantum source of entropy (a beam splitter). Comscire produces a rival product, the PQ32MU, which uses quantum shot-noise as its entropy source. Both companies provide multiple models of QRNG with varying speeds, all with appropriate statistical test results associated with their devices. As previously discussed, IDQ provides a relatively robust test report, though it is limited to Diehard and NIST SP800-22 tests. Comscire uses few and small samples, with a smaller number of tests, limiting the rigor of its test process significantly. None of the devices tested as a part of this work were validated using the AIS-31 methodology, nor were they certified (as there are no official certifications for standalone RNGs).

Data collection is not an issue from these devices, the Quantis devices provide data at a rate of 4 or 16 Mb/s, whiles the PQ32MU has an output speed of 32 Mb/s. As a result, collecting large amounts of data is trivial. A key difference in these two brands is that the Quantis generators do not implement on-board post-processing to remove bias, whilst the PQ32MU is an all-in-one product with post-processing performed on-device.

Table 10.6 shows the results for the EV1, Quantis generators and PQ32MU. Both raw and post-processed Quantis output is shown. EV1 data is tested for 64 MB samples over 100 cards. Quantis and Comscire QRNGs are tested over 100 2.1 GB

**Table 10.6** Dieharder, NIST and TestU01 results

| Device | Samples # | Dieharder passed | NIST SP800-22 passed | Alphabits passed | Rabbit passed | Small crush passed | Crush passed |
|---|---|---|---|---|---|---|---|
| DESFire EV1 | 100 | 100 | 98 | 0 | 0 | – | – |
| Quantis 16M | 100 | 100 | 100 | 54 | 60 | 93 | 47 |
| Post 16M | 100 | 100 | 100 | 95 | 87 | 91 | 82 |
| Quantis 4M | 100 | 100 | 100 | 3 | 7 | 91 | 3 |
| Post 4M | 100 | 100 | 100 | 91 | 82 | 93 | 86 |
| Quantis USB | 100 | 100 | 100 | 3 | 21 | 89 | 3 |
| Post USB | 100 | 100 | 100 | 90 | 81 | 97 | 80 |
| Comscire PQ32MU | 100 | 100 | 100 | 91 | 86 | 93 | 84 |

samples collected from one of each type of device. Ideally, more devices would be tested, but the cost was a limiting factor (the cheapest device, a 4M, costs €900).

All devices pass Dieharder, while all but 2 EV1's pass the SP800-22 tests. The TestU01 toolkit has been used, with 4 of its statistical test batteries used to evaluate all tested devices, including the EV1. Due to the sample size requirements of the Crush tests, EV1 data has not been tested for either Crush test. Immediately, the EV1 shows critical issues, failing the Alphabits and Rabbit batteries. The average failure rate is 1 of 4 tests for Alphabits, and 5 of 16 tests for Rabbit. This shows how the simple addition of a new test battery can instantly reveal weaknesses that the better-known batteries cannot identify.

Raw Quantis samples, especially those of the 4M and its USB variant, also perform very poorly on Alphabits and Rabbit. They also perform very poorly in Crush, but a significant number of samples pass the Small Crush tests. This could be because the Small Crush battery has many tests in common with SP800-22, leading to a correlation between the results. Post-processing cleans up many of these issues, but not completely. Most notably, Alphabits, Rabbit and Crush test results improve dramatically, with the most drastic change being the jump from 3 passed tests for the 4M under Alphabits, to 91 passes. This shows that appropriate use of a QRNG is yet another factor to consider: improper use of a device may not be identified by the more well-known test batteries and incorrect configuration can be as damaging as any other form of non-randomness. The Comscire PQ32MU performs well on most tests but struggles with the Rabbit and Crush tests.

Table 10.7 shows the results of Ent for the QRNG. DESFire results are not shown to avoid repetition. A summary of the 100 samples tested shows that Post-processed Quantis data, and the PQ32MU, passes the $\chi^2$ and serial correlation tests with no issues. All devices pass the other tests, hence their omission from this table. However, the raw Quantis data fails the $\chi^2$ test dramatically. Furthermore, the 4M and its USB variant perform quite poorly on the serial correlation test at the bit level. This emphasizes the need to test sequences across multiple block sizes to identify issues that may occur at lower or higher orders of output. Unlike the EV1, raw Quantis data does not provide an easily identifiable or consistent bias across

**Table 10.7** ENT results

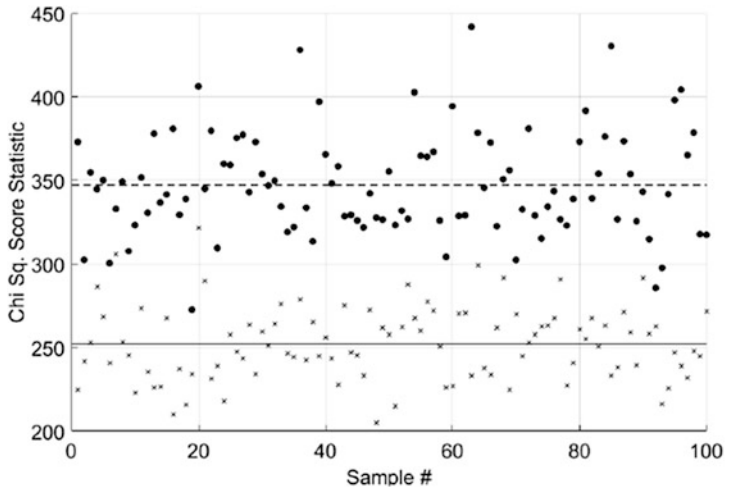
| Device | Samples # | Bytes $\chi^2$ passed | Bytes Serial corr. passed | Bits $\chi^2$ passed | Bits Serial corr. passed |
|---|---|---|---|---|---|
| Quantis 16M | 100 | 10 | 99 | 0 | 100 |
| Post 16M | 100 | 100 | 96 | 100 | 96 |
| Quantis 4M | 100 | 0 | 99 | 0 | 49 |
| Post 4M | 100 | 100 | 99 | 100 | 100 |
| Quantis USB | 100 | 0 | 92 | 0 | 81 |
| Post USB | 100 | 100 | 94 | 100 | 100 |
| Comscire PQ32MU | 100 | 100 | 99 | 100 | 100 |

samples. The bias appears to drift between samples, with the only constant being a tendency to express a $10^{-6}$ bias above the normal for byte values 0–5. Even this is not a representative trend, with only 38% of samples showing this particular trait.

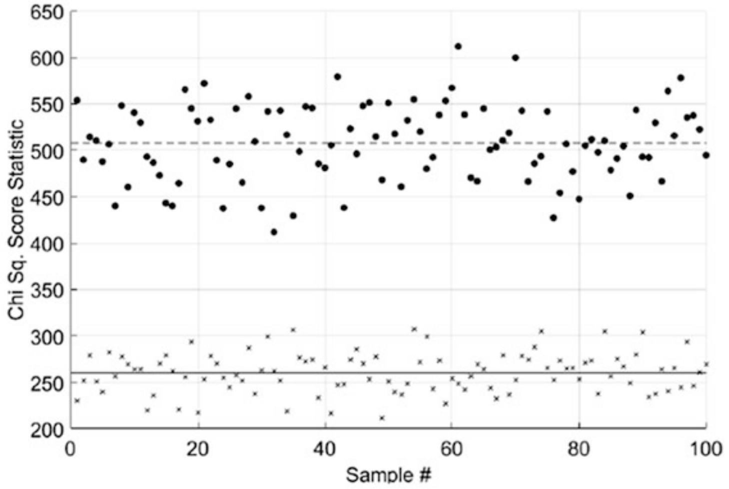
Figure 10.3 shows the $\chi^2$ statistics for raw Quantis samples from all the 16M and 4M devices. The results for the Quantis USB are omitted, as the USB is effectively a 4M in different packaging and provides similar results.

The 16M (a) fails the $\chi^2$ test for 90 of its samples. The mean statistic for the 16M is approximately 350. This is above the acceptable maximum threshold for this test. The 4M is significantly worse, with a mean statistic of 506. Unlike the 16M, the 4M shows no passes at all (the USB reports similar results). In fact, the minimum statistic for the 4M was 407. This is significantly above the maximum threshold for the $\chi^2$ test.

The experiments conducted over these QRNGs show that established tests do not always identify issues that more recent (or just less well-known) tests highlight. The TestU01 battery reinforces the results of the Ent test, by providing a wider variety of more sophisticated tests that prove that there are issues beyond simple deviation from the normal distribution of values at the byte and bit level. As TestU01 is designed to provide the tools to test TRNG, this battery would ideally be made a mandatory recommendation for TRNG and QRNG testing. Dieharder and NIST SP800-22 will remain in use, as they are effective at identifying egregious issues with RNG output, but the extension of the minimum recommended number of tests is very much needed at this time. Post-processed and raw data should be tested and the results clearly marked to show users how the improper configuration of software post-processing can be identified and resolved. One should also consider that if IoT QRNGs are sought-after, how does one implement a post-processing algorithm (which are known for their high memory requirements) in such a small package? Resource limitations may prevent effective post-processing of QRNG output, the consequences of which are made clear in the preceding work.



(a)



(b)

**Fig. 10.3** Distribution of $\chi^2$ scores for Quantis devices. (**a**) 16M. (**b**) 4M

## 10.5 Conclusion

There are many complex issues to consider when evaluating RNGs for use in security systems. Device specifications, the use of off-device entropy pools, post-processing, and output speed are all critical to the evaluation process. Each element should be tested in isolation, but only the whole device may be certified, leading to issues when considering black-box design philosophy and resource-constrained devices.

The DESFire EV1 has been found to output biased values from its TRNG, but this does not necessarily mean that the RNG itself is at fault. Subsequent work with the EV2 has found no issues with its RNG. Combined with conversations with NXP's engineers, this indicates that the issue may instead be in the whitening function employed to remove bias from the raw TRNG output. The EV1 results highlight two key issues; the role of black box design in complicating the evaluation process, and the quality control challenges facing small-scale robust TRNGs. Quantis QRNGs also require post-processing, as demonstrated by the exceptionally poor results shown by raw data over a variety of statistical tests. Rigorous testing of RNG with multiple input and processing dependencies should require results demonstrating the performance of both raw and processed output of such devices. This will aid in the identification of implementation errors.

Reliance on Dieharder and NIST SP800-22 cannot continue to the exclusion of new tests, such as those employed by TestU01. There is a wealth of academic literature on the subject of statistical tests of randomness and efforts must be made to identify which of these will provide the next wave of reliable tests of randomness. Finally, it is important to consider how tests may be evaded by manipulation of RNG output; future work will focus on how some simple manipulations result in predictable data that passes current statistical tests of randomness.