applied sciences

Article

Weaknesses in ENT Battery Design

Elena Almaraz Luengo [1], Bittor Alaña Olivares [1], Luis Javier García Villalba [1,*] and Julio Hernández-Castro [2]

[1] Group of Analysis, Security and Systems (GASS), Universidad Complutense de Madrid, 28040 Madrid, Spain; ealmaraz@ucm.es (E.A.L.); balana@ucm.es (B.A.O.)
[2] School of Computing, University of Kent, Canterbury CT2 7UL, UK; jch27@kent.ac.uk
* Correspondence: javiergv@fdi.ucm.es

Featured Application: The ENT battery presents vulnerabilities that will be shown in this work. The scope of this work is important because, in the light of the results obtained, the design of the battery tests should be reconsidered for its more effective use.

Abstract: Randomness testing is a key tool to analyse the quality of true (physical) random and pseudo-random number generators. There is a wide variety of tests that are designed for this purpose, i.e., to analyse the goodness of the sequences used. These tests are grouped in different sets called suites or batteries. The batteries must be designed in such a way that the tests that form them are independent, that they have a wide coverage, and that they are computationally efficient. One such battery is the well-known ENT battery, which provides four measures and the value of a statistic (corresponding to the chi-square goodness-of-fit test). In this paper, we will show that this battery presents some vulnerabilities and, therefore, must be redefined to solve the detected problems.

Keywords: cryptography; hypothesis testing; ENT; generators

check for updates

Citation: Almaraz Luengo, E.; Alaña Olivares, B.; García Villalba, L.J.; Hernández-Castro, J. Weaknesses in ENT Battery Design. *Appl. Sci.* **2022**, *12*, 4230. https://doi.org/10.3390/app12094230

Academic Editor: Arcangelo Castiglione

Received: 10 February 2022
Accepted: 1 April 2022
Published: 22 April 2022

1. Introduction

In many applications in different areas such as Statistics (Refs. [1–3], among others), Computational Simulation (for example, [4–6]) or Cryptography (for example, [7–9]), among others, it is necessary to work with random (or pseudo-random) sequences(A sequence of numbers is a finite ordered set $\{x_i\}_{i=0}^{n-1}$, such that $x_i \in C$ for all $i:\ 0 \le i \le n-1$, where C is a finite set of integers. Usually, $C = \{0,1\}$ or $C = \{k \in \mathbb{Z} : 0 \le k \le 255\}$. Notation by juxtaposition is allowed: $x = x_0 \ldots x_{n-1}$ [10]. A sequence of numbers $x = x_0 \ldots x_{n-1}$ (where $x_i \in C = \{0, \ldots, k-1\}$) is random if it corresponds to a simple random sample of n independent and identically distributed random variables (i.i.d.), with a discrete Uniform distribution $\mathcal{U}\{0, k-1\}$. Note that if $C = \{0,1\}, \mathcal{U}\{0,1\}$ = Bernoulli(0.5) [10]). It is of vital importance that these sequences are reliable. Imagine, for example, in a cryptographic key security application, any vulnerability in the sequence used could have catastrophic results such as the possibility of the key being discovered and the system being attacked. This is why it is necessary to check the sequences before considering their further use. The most powerful tools that can be considered for this purpose are hypothesis tests. These tests measure from different perspectives whether a sequence verifies the desired properties of uniformity and randomness. Generally, these tests are grouped in sets called batteries or suites. These batteries or suites provide the user with the possibility to perform the sequence checks in an organized and rather mechanical way. A battery or suite must fulfill some essential properties, the tests that compose them must be carefully chosen so that there are no dependent tests (which would slow down the use of the battery and would not provide more information about the sequence); furthermore, the tests that are incorporated into a battery must have a solid mathematical basis and must efficiently verify whether or not a sequence fulfills a certain property. On the other hand, a battery must have a wide coverage. There are several batteries or suites in the literature such as Diehard [11], Dieharder [12], NIST SP 800-22 [10], Cryp-X [13] or ENT [14], among others (for a more complete vision of this issue, you can see [15]).

This paper will focus on the ENT battery. As will be seen throughout this paper, this battery, which is widely used due to its simplicity and ease of implementation, has some vulnerabilities that make it unsuitable for certain applications such as cryptographic security.

2. ENT Battery

The aim of the battery is to evaluate pseudo-random number generators (a pseudo-random number generator is an algorithm that, given an input known as a seed, produces a sequence of pseudo-random numbers. The output of pseudo-random generators is usually a deterministic function with respect to its seed, so that the input must be from a random source. This is why they are called pseudo-random) for cryptographic, sampling, compression algorithms and other applications [14]. The sequence is treated at the byte level (there is also the additional option to treat it at the bit level). That is, eight bits at a time are interpreted as non-negative integers between 0 and 255, inclusive (If we represent a sequence $x = x_0 \dots x_{n-1}$ as a sequence of numbers of m binary digits (with $m|n$), we will have $x = \{x_i\}_{i=0}^{n/m-1}$, where $x_i \in \{0, 1, \dots, 2^m - 1\}$. If the considered sequence is a sequence of bytes (eight bits), the sequence is treated as numbers between 0 and $2^8 - 1 = 255$). This battery provides five results corresponding to the calculation of four measures and the value of the chi-square test statistic: entropy, chi-square, arithmetic mean, Monte Carlo estimation of π and serial correlation. In what follows, we will describe them briefly.

2.1. Entropy

The battery gives the entropy as conceived in information theory [16]. Let the sequence of bytes be denoted $x = \{x_k\}_{k=0}^{n-1}$. The occurrences of each number i between 0 and 255 in the sequence are counted, and their frequencies are computed to consider the probability that each number has. In this way, it is said that $P_x(i) = \frac{Ap_i}{n}$, where $Ap_i = \#\{0 \leq k \leq n-1 : x_k = i\}$. Next, the expectation of the logarithm in base two of one divided by the probability of each number is found [16]:

$$H(x) = -\sum_{i=0}^{255} P_x(i) \log_2 P_x(i). \tag{1}$$

Intuitively, entropy tells how many bits of incompressible information each byte of the sequence contains. Thus, running the test on a sequence gives a value between 0 and 8, where 0 means that the sequence does not contain any information, while 8 means that the sequence is noncompressible. In turn, it is shown on the screen how compressible the sequence is, which is obtained by the following fraction:

$$100 \times \frac{8 - \text{entropy}}{8}.$$

Therefore, a sequence with 8 bits per byte of entropy is 0% compressible, while a sequence with 4 bits per byte will be 50% compressible. Note that to achieve perfect entropy, the numbers must be uniformly distributed, which will result in $P_x(i) = 2^{-8}$ for each $i \in \{0, 1, \dots, 255\}$, so that $H(x) = -\sum_i 2^{-8}(-8) = 8$. It is easy to see that this test could be fooled, and biased sequences could present a good value for this measure. That is, it is enough to have a counter from 0 to 255 cyclically to obtain a perfect statistic on this measure.

2.2. Chi-Square

The χ^2 test measures the goodness-of-fit with respect to the number of occurrences that can be expected for each number in a byte under the assumption of uniform randomness. Let us refer to the statistic it obtains as X^2, which has the following form [14]:

$$X^2 = \sum_{0 \le i \le 255} \frac{(Ap_i - ApT_i)^2}{ApT_i} \tag{2}$$

where Ap_i are the occurrences of the number i in the sequence and ApT_i are the theoretical occurrences under randomness assumption. In other words, $Ap_i = \#\{0 \le k \le n-1 : x_k = i\}$ and $ApT_i = \frac{n}{256}$. In addition, it gives as a percentage a p-value, that is, how likely it is to obtain the seen or a more extreme (larger) output under the randomness assumption: $p\text{-value} = P(\chi^2_{255} \ge X^2)$, where χ^2_{255} is the chi-square distribution with 255 degrees of freedom. As it can be seen, this test focuses, again, on whether the occurrences are well-balanced.

2.3. Arithmetic Mean

It is nothing else than the average of all bytes interpreted as numbers between 0 and 255, with an expectation of 127.5 under randomness. It is denoted by $\overline{x} = \frac{1}{n}\sum_{i=0}^{n-1} x_i$.

2.4. Monte Carlo Estimation of π

To perform this test, coordinates are taken from the plane in the rectangle $[0, 2^{24}-1]^2$. To take a point, three bytes are used for the abscissa x and another three for the ordinate y. As 3 bytes correspond to 24 bits, they represent integer values between 0 and $2^{24}-1$. Having found the pair (x, y), it is wanted to see if it is inside the closed Euclidean ball of the center $(0, 0)$ and radius $r = 2^{24} - 1$. This is true if and only if $x^2 + y^2 \le r^2$.

The points inside the ball and the total number of points are counted. With the quotient between the coordinates inside the circle and the total number of coordinates, the division between the area of the circle inside the rectangle and the area of the rectangle is approximated. This is given by:

$$\frac{\text{points inside the circle}}{\text{total points}} \simeq \frac{\text{area of the partial ball}}{\text{area of the rectagle}} = \frac{\pi \cdot r^2/4}{r^2} = \pi/4. \tag{3}$$

Then, the estimation of π is obtained by $\tilde{\pi} = 4 \cdot \frac{\text{points inside the circle}}{\text{total points}} = 4 \cdot \frac{\text{points}_{\text{in}}}{\text{points}}$. In addition, the percentage error of the estimation is given.

2.5. Serial Correlation

We measure how much each byte depends on its immediately preceding byte. It counts the occurrences of each pair of bytes and returns a value between -1 and 1, where 0 means that the bytes are totally uncorrelated and an absolute value of 1 would mean that they are totally correlated.

This coefficient is obtained through the Pearson's correlation coefficient r between the sequence $x = x_0 \ldots x_{n-1}$, and the sequence shifted one byte "to the left" $x' = x_1 \ldots x_{n-1}x_0$.

Recall that the sample correlation coefficient for two random samples x, y, with sample means $\overline{x}$ and $\overline{y}$, respectively, is given by the following expression:

$$r = \frac{\sum\limits_{i=0}^{n-1}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum\limits_{i=0}^{n-1}(x_i - \overline{x})^2}\sqrt{\sum\limits_{i=0}^{n-1}(y_i - \overline{y})^2}}. \tag{4}$$

which, in this case with x and x', results in:

$$r = \frac{\sum_{i=0}^{n-1} x_i x_{(i+1) \bmod n} - n\overline{x}^2}{\sum_{i=0}^{n-1} x_i^2 - n\overline{x}^2}$$

As explained, only one of the measures is a test and has a p-value. This means that the interpretation of the remaining statistics is not firmly established.

3. Experimentation Design and Results

Due to the weakness of the ENT battery, it is possible to conceive of deliberately biased sequences which, on the contrary, are able to provide good test and measures results. To demonstrate this, in this experimentation, sequences generated from biased generators will be used. These generators were designed in [17], and their designs were made in such a way as to introduce biases at varying levels into an apparently random sequence, which will then be checked by the tests, in this case, of the ENT battery. From the results obtained, it will be possible to deduce whether the battery tests are sensitive enough to detect these biases and reject the sequence. Specifically, the generated biases that will be used are AES σ-counter, ϵ-hole and t-counter.

The parameters σ, ϵ and t will indicate how much bias is introduced into sequences between 0 and 1.

These generators have very apparent biases, and it would not be appropriate to use them for encryption purposes, as these biases can be used against them for malicious purposes, such as brute force attacks.

To test what thresholds the ENT battery is or is not capable of detecting, a wide range of parameters will be considered. The three generators are parameterized, respectively, by σ, ϵ and t. For the experiment shown in this paper, 21 parameter values are taken, varying them from 0 to 1 (both inclusive) with 0.05 jumps. For each parameter value, 100 files are taken, resulting in 2100 files for each generator. To unify the nomenclature of the parameters in the experimentation, they will be denoted by θ. The file sizes are 13 MB, 10 MB and 1 MB, respectively. In our notation, we will refer to $M = 10^6$, as is the usual convention in science, not the 2^{20} that would be in computer science. To have a benchmark against which to test the biases, we will use 10,000 files from */dev/urandom*.

The ENT battery will be run for all generated data, and the results will be compiled for further interpretation.

The biased generators used in the experimentation will be briefly described below.

3.1. AES σ-Counter

First, a sequence $S = S = \{S_i\}_{i=0}^{255}$ is fixed, which is a permutation of $\{0, 1, 2, \ldots, 255\}$ (understood as an ordered set) and apparently random, achieved by encrypting the latter set by means of AES. Then, and keeping a circular counter from 0 to 255 (which goes from 255 to 0), the sequence is obtained as follows. Let the i-th element of the sequence be denoted by X_i, and let R_i be a random byte obtained using the generator of the computer:

$$X_i = \begin{cases} S_c & \text{with probability } \sigma \\ R_i & \text{with probability } 1-\sigma \end{cases} \tag{5}$$

whenever S_c is drawn, c is incremented, $c' = (c+1) \mod 256$. Thus, the counter is "filtering" the sequence set with probability σ. When $\sigma = 1$, the output of the generator is S repeated in a loop.

3.2. ϵ-Hole

This generator creates a "hole", eliminating occurrences of a certain number. Let b be that number. For the experiments, b has been taken to be $b = 255$. Then, the output is:

$$X_i = \begin{cases} R_i \backslash \{b\} & \text{with probability } \epsilon \\ R_i & \text{with probability } 1 - \epsilon \end{cases}$$

3.3. t-Counter

This generator has somewhat more subtle differences than the previous ones. Using the same fixed sequence S as before, the output of the generator is either a random number, with probability $1 - t$, or a S_y, where the index y will be "jumping" between close values of the index used in the previous use of S, which will be called i. That is to say, this would give:

$$y = i + r \mod 256, \quad r \in [-n, \ldots, 2n] \text{ random.}$$

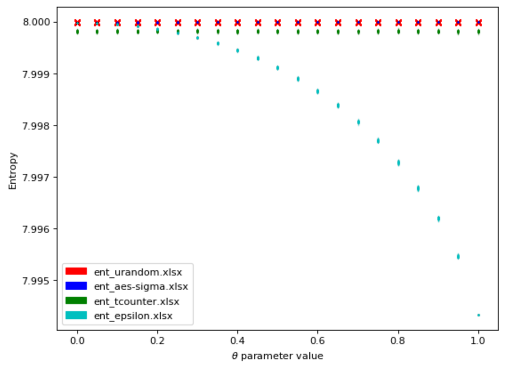
The indices will change back and forth, but always within a small "window" of fixed size $3n + 1$, centered on the previously observed index. We will always have that $|y - i| \leq 2n$. For our generation, we have used $n = 20$. So:

$$X_i = \begin{cases} S_y & \text{with probability } t \\ R_i & \text{with probability } 1 - t \end{cases}$$

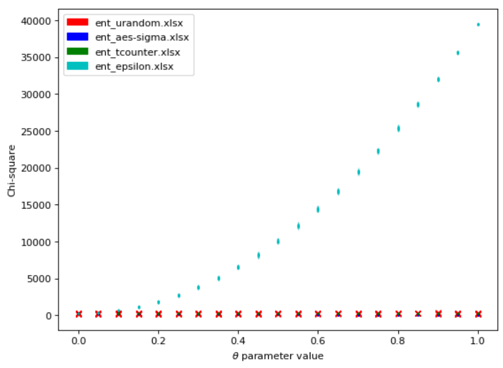
3.4. Results

In Figure 1, the results obtained with the different sequences generated are represented. The data generated by */dev/urandom*, not having a parameter on which they depend, have simply been divided into 21 groups in the order in which they were tested, so that they can be used for reference and comparison.

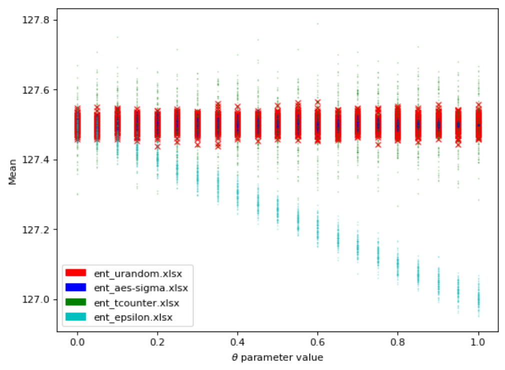
As it can be seen, in Figure 1a, all generators have good entropy data in principle, except for the ϵ-hole. This is very intuitive: As the θ parameter grows, the more likely it is that the 255 byte in the sequence will be missed. Finally, when θ is 1, there are no 255 s in the sequence, and so the numbers are not well distributed and entropy drops considerably. Note that even so, the lowest values when $\theta = 1$ are on the order of 7.99, despite being comparatively disastrous compared to random data (*dev/urandom*), or evenly distributed data (albeit sequentially), such as the AES σ-counter case. The Figure 1b reflects the same trend as above, as 255 disappears from the sequence, the summand of Equation (2) belonging to $i = 255$ shoots up, having $Ap_{255} = 0$. This is reflected in low p-values for this generator. In fact, for $\theta = 0.05$, 91 out of 100 ϵ-hole sequences have a p-value less than 0.01. This shows that ENT detects the *epsilon*-hole bias very easily. Figure 1c also shows the bias of ϵ-hole, which obviously sees its arithmetic mean significantly reduced as the number of occurrences of 255 decreases. In addition, this graph also shows that the t-counter sequences take quite alternating values and are quite distant from 127.5. However, looking at the glaring difference with the data from *dev/urandom*, everything points to the fact that the means of these generators are not in an acceptable range. Meanwhile, Figure 1d shows that both the σ-counter and ϵ-hole obtain values with large error as their parameter grows. Finally, Figure 1e reveals that for both the t-counter and the σ-counter, the sequence depends heavily on the previously observed values. In fact, if we look at the expression (5) that defines the sequence, it can be seen that when $\sigma = 1$, the sequence is just a periodic repetition of S. This is reflected in how the values are triggered in Figure 1e. However, we note that the highest value observed is 0.015. It is therefore not straightforward to determine at a glance how much correlation is acceptable, and how the length of the sequence affects the threshold to be chosen. Therefore, it would also be useful to have a p-value associated with this test.



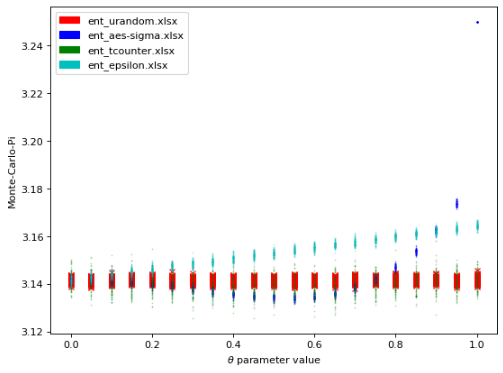
(**a**) Entropy



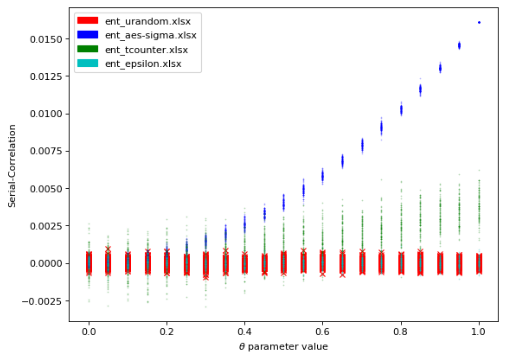
(**b**) Chi-square



(**c**) Arithmetic Mean



(**d**) Monte Carlo estimation of π



(**e**) Serial Correlation

Figure 1. Result of the statistics for each generator and the values of θ.

4. Conclusions

As we have seen, it is possible to define generators that produce certain biases in the sequences that are capable of deceiving the ENT battery. Thus, it has been seen how it is possible to vary the values of the θ parameter, and even if deliberately biased sequences are generated, it is not possible for the ENT battery tests to detect such biases. In particular, for small values of θ (read 0–0.2), all biased generators provide sequences that show very similar results to those that would result from applying the battery to sequences generated by *dev/urandom* of known quality. This is why we do not consider this battery suitable as a testing tool, especially for cryptographic applications, where security is a fundamental requirement. We recommend a review of the tests that make it up.

Author Contributions: Conceptualization, E.A.L., B.A.O., L.J.G.V. and J.H.-C.; methodology, E.A.L., B.A.O., L.J.G.V. and J.H.-C.; software, E.A.L., B.A.O., L.J.G.V. and J.H.-C.; validation, E.A.L., B.A.O., L.J.G.V. and J.H.-C.; formal analysis, E.A.L., B.A.O., L.J.G.V. and J.H.-C.; investigation, E.A.L., B.A.O., L.J.G.V. and J.H.-C.; data curation, E.A.L., B.A.O., L.J.G.V. and J.H.-C.; writing—original draft preparation, E.A.L., B.A.O., L.J.G.V. and J.H.-C.; writing—review and editing, E.A.L., B.A.O., L.J.G.V. and J.H.-C.; visualization, E.A.L., B.A.O., L.J.G.V. and J.H.-C. All authors have read and agreed to the published version of the manuscript.

Funding: This research work has received funding from UCM Projects THEIA (FEI-EU-19-04), THEIA I (FEI-EU-21-01) and THEIA II (FEI-22-01). JHC is supported by EPSRC's Quantum Communications Hub (grant number EP/T001011/1) and Innovate UK Industrial Strategy Challenge Fund (ISCF) Project No. 106374–49229 AQuRand (Assurance of Quantum Random Number Generators).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We thank the reviewers for their comments and indications that have helped to improve this work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wang, L.; Cheng, H. Pseudo-Random Number Generator Based on Logistic Chaotic System. *Entropy* **2019**, *21*, 960. [CrossRef]
2. Figueroa-García, J.C.; Varón-Gaviria, C.A.; Barbosa-Fontecha, J.L. Fuzzy Random Variable Generation Using α-Cuts. *IEEE Trans. Fuzzy Syst.* **2021**, *29*, 539–548. [CrossRef]
3. Cotrina, G.; Peinado, A.; Ortiz, A. Gaussian Pseudorandom Number Generator Using Linear Feedback Shift Registers in Extended Fields. *Mathematics* **2021**, *9*, 556. [CrossRef]
4. Gergely, A.M.; Crainicu, B. A succinct survey on (Pseudo)-random number generators from a cryptographic perspective. In Proceedings of the 2017 5th International Symposium on Digital Forensic and Security (ISDFS), Tirgu Mures, Romania, 26–28 April 2017; pp. 1–6. [CrossRef]
5. Wang, P.; You, F.; He, S. Design of Broadband Compressed Sampling Receiver Based on Concurrent Alternate Random Sequences. *IEEE Access* **2019**, *7*, 135525–135538. [CrossRef]
6. Gómez, A.I.; Gómez-Pérez, D.; Pillichshammer, F. Secure pseudorandom bit generators and point sets with low star-discrepancy. *J. Comput. Appl. Math.* **2021**, *396*, 113601. [CrossRef]
7. Hurley-Smith, D.; Hernández-Castro, J. Certifiably Biased: An In-Depth Analysis of a Common Criteria EAL4+ Certified TRNG. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 1031–1041. [CrossRef]
8. Lee, S.; Jho, N.S.; Chung, D.; Kang, Y.; Kim, M. Rcryptect: Real-Time Detection of Cryptographic Function in the User-Space Filesystem. *Comput. Secur.* **2021**, *112*, 1–26. [CrossRef]
9. Tang, J.; Jiao, L.; Zeng, K.; Wen, H.; Qin, K.Y. Physical Layer Secure MIMO Communications against Eavesdroppers with Arbitrary Number of Antennas. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 466–481. [CrossRef]
10. Rukhin, A.L.; Soto, J.; Nechvatal, J.R.; Smid, M.E.; Barker, E.; Leigh, S.; Levenson, M.; Vangel, M.; Banks, D.; Heckert, A.; et al. *SP 800-22 Rev. 1a. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*; Technical Report; Special Publication (NIST SP); National Institute of Standards and Technology: Gaithersburg, MD, USA, 2010. Available online: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=906762 (accessed on 9 February 2022).
11. Marsaglia, G. *The Marsaglia Random Number CDROM Including the Diehard Battery of Tests of Randomness*; National Science Foundation: Alexandria, VA, USA, 1995.
12. Brown, R.G.; Eddelbuettel, D.; Bauer, D. Dieharder: A Random Number Test Suite (Version 3.31.1). 2014. Available online: https://webhome.phy.duke.edu/~rgb/General/dieharder.php (accessed on 9 February 2022).
13. Gustafson, H.; Dawson, E.; Nielsen, L.; Caelli, W. A computer package for measuring the strength of encryption algorithms. *Comput. Secur.* **1994**, *13*, 687–697. [CrossRef]
14. Walker, J. ENT: A Pseudorandom Number Sequence Test Program. 2018. Available online: https://www.fourmilab.ch/random/ (accessed on 9 February 2022).
15. Almaraz Luengo, E.; García Villalba, L.J. Recommendations on Statistical Randomness Test Batteries for Cryptographic Purposes. *ACM Comput. Surv.* **2021**, *54*. [CrossRef]
16. Gray, R.M. *Entropy and Information Theory*, 2nd. ed.; Springer: New York, NY, USA, 2011.
17. Hurley-Smith, D.; Patsakis, C.; Hernández-Castro, J. On the unbearable lightness of FIPS 140-2 randomness tests. *IEEE Trans. Inf. Forensics Secur.* **2020**, 1–13. [CrossRef]