Entropy in the Physical World

Ronnie Blondale, Emma Lubes, Patrick Marchione, Loudon Mehling

***Abstract*—Cryptographic algorithms need sufficiently random numbers to generate keys and maintain security. These numbers are currently generated via Pseudo Random Number Generators (PRNGs) and random enough to not allow attackers to correctly determine the random numbers. Yet as more Internet of Things (IoT) devices and embedded systems, which lack the computation power to generate sufficient random numbers, increase, the security of these devices decrease. This study aims to identify potential physical sources of entropy, or randomness, that could be used to generate random numbers that can be used in devices such as IoT and embedded systems. The potential sources' datasets were identified and collected through Google Dataset Search along with RealRandom, LLC. providing their own entropy server, Entropy As A Service (EAAS) and was then tested using the NIST Statistical Testing Suite. The results showed that the EAAS server performed the best against the NIST STS tests, almost 3.25 times better than the second best, barometric pressure. We conclude that entropy derived from physical sources should be used in cryptographic algorithms as PRNGs generate insufficient random numbers and introduce vulnerabilities to the systems. In the future, generators should be built for certain physical sources such as wind speed, wind direction and barometric pressure instead of using static datasets.**

***Index Terms*—entropy, random number generation, security, physical sources, weather entropy, thermal noise entropy, NIST Statistical Test Suite**

I. Introduction

True randomness is necessary for many elements of cybersecurity. The majority of these use cases lie within cryptography and authentication [1]. Individually, these use cases include encryption keys, MAC algorithm keys, digital signature keys, authentication mechanism values, key establishment values, PINs, passwords, and nonces [2], [3]. These algorithms rely on true randomness to remain as secure as possible.

The keys created and used within cryptographic algorithms can become predictable and insecure without true randomness. As a whole, cryptographic algorithms are only as strong as the keys used within them are random [4], [5], [6], [7]. Attackers can mine predictable keys to find common primes used for seeding their generation algorithm [5]. Users then believe their device is secure because they use a secure encryption algorithm. In reality, their device is insecure because of insufficient randomness used within the algorithm, of which they have no control. Not only do users not have control over the source of randomness used to seed their encryption algorithms, but companies also use no single systematic solution to mitigate vulnerabilities identified to be caused by poor entropy [4]. Therefore, a device using poor entropy may not be entirely secure, leaving a user insecure due to an algorithm they believed to be securing them.

Algorithms used to generate RSA keys and other forms of cryptographic keys are deterministic; in order to create different keys, the algorithm must be supplied with a seed to generate a key [2]. Every time a key is generated with that seed, it will be the same. If an attacker can guess the seed used to generate a key, they too can generate the same key. For example, suppose a program generates a key using the current time as the seed, and an attacker can estimate the generation time roughly. In that case, they will have to try significantly fewer combinations than a traditional brute force attack.

Using unpredictable, random seeds can mitigate this attack [4]. The current mitigation for this attack is to use Cryptographically Secure Pseudo-Random Number Generators (PRNGs). In most cases, they are random enough that an attacker cannot narrow down the number of seeds to try. However, there has been a recent increase in IoT and embedded systems with minimal computing power, and the PRNGs on these devices are extremely limited. [5]. Therefore, they cannot create a wide enough range of seeds, allowing attackers to easily reverse engineer these weak keys.

The recent milestone of quantum supremacy also brings to light the need for quantum-resistance encryption in the post-quantum world. Classical True Random Number Generators (TRNGs), which are not based on deterministic algorithms and easily-solvable quantum problems, such as factoring, will be significantly harder for quantum computers to break [8]. It is of upmost importance that cryptographic keys are generated with sufficient entropy as the world begins to use quantum computing. Allowing for more direct access to an entropy source, such as using a physical source of entropy, may also aid in the creation of difficult-to-break quantum security.

The proposed solution to this issue of insufficient randomness on computationally limited devices is what is referred to as an entropy authority. An entropy authority would address this issue by providing externally generated random data to these devices [4]. This data will be provided in the form of random bits. From there the devices in question would be able to generate keys for secure communication and remain secure. Since the entropy authority is an external entity, it can devote an extreme amount of computing power to generating or collecting random data. The communication process between a device and an entropy authority is depicted in Figure 1. Though the scheme is clear, the actual source of random data is not outlined. As such this project aims to assess the viability of various sources of data for an entropy authority.

Unlike non-physical sources derived from deterministic algorithms, physical sources of entropy can provide true randomness [9], and will be used by the entropy authority. Physical chaotic sources use pre-existing uncertainties in conditions to generate randomness. Because physical sources of entropy are based on uncertainities already existing in the system, they cannot be reversed by an attacker to determine the next number in the sequence. However, PRNGs suffer from this predictability, as they are completely deterministic algorithms.

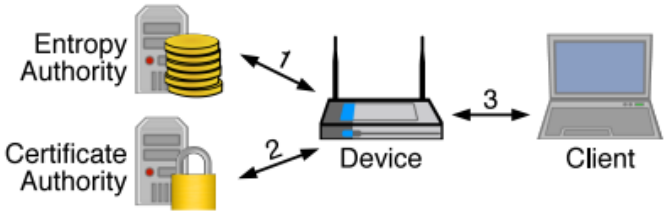

Fig. 1. The entropy authority infrastructure [4].

Therefore, physical sources of entropy can be used to seed PRNGs and create unbiased, uniformly distributed sequences of random numbers.

Our project is a sub-project of the blockchain entropy authority. Our project goals include identifying potential physical sources of entropy to be used by entropy authorities, locating APIs for these sources, and analyzing the performance of the data within the NIST Statistical Test Suite. Through this project, sample data from the real random entropy as a service server, thermal noise, wind direction, wind speed, humidity and barometric pressure were analyzed. Overall, the project found that the data provided by the entropy as a service server was of the highest quality by far. The other data-sets analyzed were of poor quality, all falling below 30 percent combined pass rate through the NIST statistical testing suite.

II. Background

A. The Importance of True Randomness

1) Properties of True Randomness: A source of randomness has multiple properties that it must fulfill to be truly random. A source must be intrinsically random in order to be truly random; the source must have an integral property that is random, instead of unpredictable external events caused by the source [2]. The most frequently used principles of entropy in source generation are unpredictability, a lack of patterns within the source-generated bit sequences, and an even distribution of bits within the outputted bit sequences [6]. Figure 2 provides a visual depiction of patterns within sequences generated by PRNGs. Researchers used these principles to generate questions to determine the suitability of a selected source for entropy analysis:

1) ”How much ’unpredictability’ does my source generate?”
2) ”How do I translate this unpredictability into random bit streams or numbers?”
3) ”How efficient is this translation?”
4) ”Is the resulting data biased or not?”
5) ”Do I get sufficient unpredictable output for my purpose?”
6) ”Are all the output bits I am getting unpredictable, or only a portion?” [2].

An entropy source that can provide adequate answers to these questions is a candidate for true randomness and should be further evaluated for the strength of its entropy.

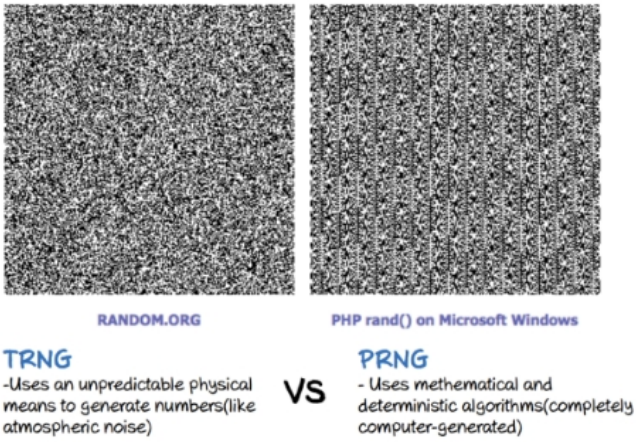

Fig. 2. Visual depiction of true randomness versus pseudo-randomness [10].

2) Obtaining and Evaluating True Randomness: Physical sources of entropy are the best candidates for obtaining true randomness. Computers are not typically used as potential sources of true randomness because engineers designed them to be as predictable as possible [6]. The most secure deterministic algorithm-based PRNGs will use physical sources of entropy to seed their algorithms [9]. A truly random source will consist of a physical noise source, data sampling, and any necessary (yet minimal) post-processing [6]. All origin points should remain separate from any post-processing that occurs to the obtained data, even though the post-processing procedures are less likely to contaminate the data than a deterministic algorithm [9], [6]. Physical sources of entropy are the best candidates to fulfill these requirements as they are not designed to be predictable and contain the necessary elements for intrinsic randomness while remaining separate from the processing mechanisms.

Researchers must evaluate physical sources of entropy for the strength of randomness they output to create secure Random Number Generators (RNGs). The current process for this includes collecting data from the physical source, conditioning the data in order to remove bias, and running a test suite on the output bit sequence [9]. Common entropy suitability test suites include the NIST Statistical Test Suite, DieHARD Battery of Tests, ENT Test Program, and Comparative Test Parameters [11]. Of these tests, DieHARD and NIST are the standards. Testing physical sources of entropy ensures that the RNGs they seed are truly random and not pseudo-random.

B. Evaluating Data for Randomness

DieHARDER is an extended test suite based on the DieHARD Battery of Tests aimed at verifying the randomness of random number generators. The NIST test suite, outlined in NIST SP 800-22 Rev 1a, has the same intention, but for both random number generators and pseudorandom number generators [12]. The DieHARD family of tests explicitly focus on the generator aspect of randomness, while the NIST test suite allows for the randomness of outputted data to be

corroborated as random. The DieHARD and DieHARDER tests suites should not be used to check the randomness of standalone data because of their focus on the generator aspect of randomness. One should note that verifying the randomness of standalone files of potentially random data itself does not ensure the randomness of the generator as a whole; one must look at the input of the generator and its intrinsic characteristics, as well as the randomness of its output, to demonstrate the randomness of the entire system [13].

The DieHARD family of test suites also requires a very large amount of output data in order to run the tests since they were developed on receiving RNG output as test input. Brown recommends at using least 10 MB of data to limit the amount of data rewinds the test must perform [13]. The NIST test suite can be used on smaller data sets and consists of 15 tests. Entropy is given a 0-10 score out of 10 based on the p-value selected at test run-time; an 8 or higher is considered passing [12].

The Frequency (Monobit) Test checks the proportion of zeroes and ones from the sequence provided to ensure that there is an equal probability that it is either a zero or a one [14]. Frequency Test within a Block tests to see frequency of ones specified in a block should equal 1/2 of the total amount of digits inside of the block. Cumulative Sums Test is used to determine if the cumulative sum of part of the sequence provided is in line with what is expected for a random sequence. The Runs Test checks to see the number of identical bits in a continuous sequence are uniform to what is expected from a random sequence and determines if there is a too small or too large of a frequency among ones and zeroes in a sequence. The Longest Run Tests determines if a block of the sequence has an expected number of continuous ones for a random sequence. The Binary Matrix Rank Test determines if a sequence broken down into sub-sequences is linearly dependent. Discrete Fourier Transform Test checks if there repetitive patterns in close proximity of one another that would deviated from what is expected from a random sequence. Approximate Entropy Test checks the frequency of overlapping blocks for the number of patterns expected from a random sequence. The Serial Test checks the number of times that 2^m patterns occur compared to that expected from a random sequence. Finally, the Linear Complexity Test determines if the length of linear feedback shift register (LSFR) is complex enough to be considered random .

III. Physical Entropy Source Candidates

The primary source themes discussed in this background were taken from 16 papers written about physical sources of entropy. They can also be found in Table I. Other potential sources considered, but whose validity could not be verified by an academic source, and will not be discussed, include harmonics, production processes, production fluctuations, consumption rates, loss rate in transmission, precipitation, solar radiation, and atmospheric RF noise.

Today, a standard piece of technology in hardware is Field Programmable Gate Arrays (FPGA), which allows non-manufacturers to program or reconfigure specific parts of a circuit. These re-programmable circuits can be used in conjunction with ring oscillators as a source of entropy through time jitters [15]. Time jitters, or delays from a signal inside a circuit, are caused by many different events such as thermal or electrical noise and interference. Non-deterministic processes inside or outside of the circuit can cause these jitters in the system. Through measuring these time jitters, the output is used for random bit generation.

Metal-oxide semiconductors (MOS) are fundamental pieces of technology used across devices today and could be a potential source of entropy. When a MOS device experiences a phenomenon called soft breakdown, large fluctuations occur due to the leakage of current through the device [16]. These large fluctuations of current caused by soft breakdown are instantaneous and unpredictable. Yasuda *et al* created a system of MOS devices and purposefully made these devices encounter soft breakdown to measure the fluctuations of the current. They found that these fluctuations are six times larger than thermal noise and can produce high-quality entropy.

Wind may also be an excellent physical source of entropy. Kim *et al* found that measuring the flow of wind is intrinsically chaotic, unpredictable, and non-reputable [17], [18]. They have created a standalone TRNG device that creates random bit generation based on measuring wind flow. They achieved this through a triboelectric energy generator (TENG) which converts the wind-flow signal to an electrical signal for the output. This methodology is a feasible, ecological, and cost-efficient solution as a source of entropy.

Currently, many devices have on-board sensors that measure the current temperature inside and outside their respective systems. Pawlowski *et al* uses an integrated temperature sensor from an IoT device to determine if the temperature can be a source of entropy [19]. They create an interface to obtain the temperature measurements from the IoT device, transfers them to an analog signal, and then returns the least significant bits of data. To generate the entropy source, they concatenated the least significant bits of data returned from the IoT device. While this study was specifically about IoT sensors, their method of extracting temperature measurements also applies to other device types.

Like temperature, sensors in aviation devices such as gyroscopes, accelerometers, and barometers are also entropy sources. Research done by Cho *et al* used these sensors for random bit generation. Similarly to [17], [18], Cho *et al* uses both the least and most significant bits returned by these sensors from a drone in a stationary and moving state [20]. While a drone was the only target device for this study, it proves that these sensors create quality entropy and scale to larger aviation devices.

Cryptocurrency mining, most notably the Bitcoin blockchain, can be used as a source for entropy. Research completed by Bonneau *et al* implemented a blockchain beacon that uses the headers of one or more Bitcoin blockchain blocks[21]. These blockchain headers are a 640-bit data structure including the block hash and fields such as the nonce and the version number previous block hash. The implemented beacon must also satisfy these security properties: to be unpredictable, to be unbiased, to

be universally sample-able, and to be universally verifiable. Bonneau *et al* suggests using beacons for smart contracts, securing election protocols and potentially other use.

Several schemes generate random numbers on IoT devices locally to make up for their inability to do so with pseudo-random means. These dedicated circuits will generate random numbers via physical phenomena, ideally making them unpredictable. Schemes include temperature, humidity and light sensors[19], SR Latches[22], and SiN MOSFET noise[23].

Spontaneous Emission is a phenomenon that occurs where an atom or specific ion releases energy that a photon will absorb. Williams *et al* use this phenomenon as a noise source for random bit generation with a laser and an amplifier [24]. The amplifier in this design creates an amplified spontaneous emission noise in the system, which is then recorded by two photoreceivers independently. With this scheme, Williams *et al* show that amplified spontaneous emission noise is much greater than that electronic noise as they have achieved a random bit generation rate of 12.5 GB/s.

Broadband optoelectronic devices emit or detect light at high speeds. Adding a chaotic optical signal within these devices causes random amplitudes and time positions which are attractive values for random bit generation [11]. Through the use of semiconductor lasers, optical chaos is created based on either optical or electro-optical feedback in the system [25]. Once chaos has been introduced, an oscilloscope converts the analog signal to a digital one, and then post-processing techniques are used to create the random bit generation [11].

There are many different methods of producing random bit generation based on a technique known as photon time-of-arrival detection. One variation of this technique is to count the number of photons in a given time interval. Hart *et al* developed a similar system where a continuous wave laser outputs photons at high speeds, which a photon-counting device records [9]. Once recorded, Hart *et al* digitizes the interarrival times between the photons and are used as the signal for random bit generation.

Finally, the last source candidate explored in this paper is the chaos created in electrical circuits. The chaos in electrical circuits allows for non-deterministic outputs through noise or changes in the configuration of the circuit, such as FPGAs [15]. Additional elements in FPGA systems can create random bit generation as well. Because electrical chaos is relatively easy to create, there are different ways to achieve random bit generation like SR Latches [22], Sin MOSFET [23], and through Ink Jet printers [26].

IV. Project Methodology and Constraints

The overall process used for the project is outlined in Figure 3. Additional details are provided in the following subsections.

A. Finding Data Sets

One of the entropy sources used for testing was the Entropy As A Service (EAAS) server provided by Real Random, LLC. This server provides 256 and 512 bits of data per request. Other Application Programming Interfaces (APIs) were considered for testing entropy sufficiency, but challenges arose that limited these APIs from being used. The main challenge of attempting to use APIs was the slow data rate. The time to gather the necessary data to run the DieHARDER tests correctly would have taken multiple days. Another challenge in using APIs was the small amount of data that a single API request sent back. The API requests would return integers verifying from one digit to eight digits. Making the necessary requests to the API to run the DieHARDER suite would be complex due to the APIs potentially blacklisting our IP address or hard-capping the number of API requests allowed, either by rate-limiting or a financial obligation.

Due to these challenges, we selected other static data sets for entropy testing. Only data sets with data that represented categories from the source list (Table I) and were at least 10 MB in size were selected. We found data sets by searching the Google Dataset Search tool for big data. Our search terms were our source candidates from Table I. We chose data sets representing wind direction, wind speed, barometric pressure, humidity, and thermal noise from the list of sources due to their availability online, and the ability to find at least 10 MB

TABLE I
Source Candidates and Categories

Category	Source Candidates
Hardware TRNG Devices	Time Jitter in Electrical Oscillators
Chemical Reaction	MOS Soft Breakdown
Weather Sensing Networks	Wind Speed and Direction, Temperature, Humidity, and Barometric Pressure
Cryptocurrency Mining	Bitcoin Mining
IoT Ecosystem Monitoring	On-Board Sensors in IoT Devices
Optical	Amplified Spontaneous Emission, Optoelectrical Chaos, and Single Photon Time-of-Arrival Detection
Stochastic	Chaotic Electrical Circuits

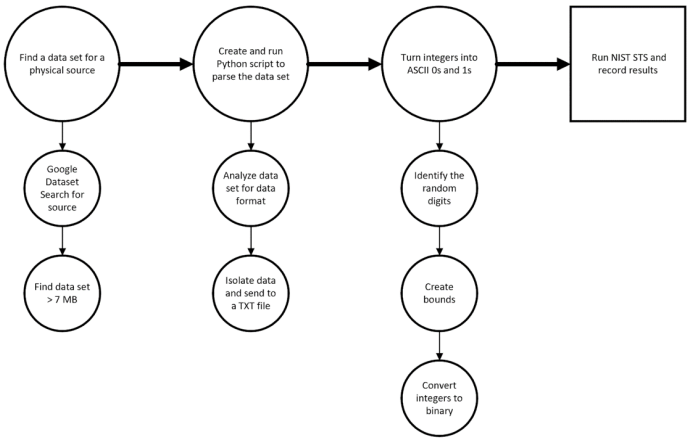

Fig. 3. A visual depiction of the data selection and testing process developed and used during this project.

of data on each source.

B. Testing Data Sets

1) Converting ASCII to Binary: For this project, potentially random testing data must be binary. Our fundamental assumption is that of this data, every individual bit has an equal chance of being a 1 or 0. However, some of the selected data sets stored data in ASCII instead of binary. As a result, this data had to be translated into binary, and maintaining statistical randomness across this translation is not straightforward. First, we must determine what part of the data we are assuming to be random for these data sources. Typically, the data will present itself in a uniform format, so assumptions of randomness will carry on for all data points.

An example of these assumptions is the testing performed with wind direction. The located data set gave readings of an angle between zero and 360 degrees. We assume that each degree measurement between zero and 360 is equally likely to appear within the data set. With different data sets, one can assume that a specific range of digits will be random, but in this case, this approach does not make sense because of the bounds. Since all angles are assumed to be equally possible, the angle measurement was converted to binary, creating a range from 000000000 to 101101000. This data cannot simply be written to a file, as the process that created the data will skew its statistical properties. The range does not reach 1s entirely, which violates the assumption that all binary bits are equally likely to be one or zero. The range must be limited in order to fix this problem. The range was limited from zero to 255 so that the represented range would be 00000000 to 11111111, and any higher data points were thrown away. This subset should also have an equal probability that any number between zero and 255 should appear, similar to how the original data set had an equal probability that any number between zero and 255 would appear. This full range should have an equal chance of all bits being 1s or 0s. From there, the bits are written to a file and tested. This process of finding the random digits, creating bounds, and converting ASCII integers to binary was used for all selected data sets.

2) The NIST Statistical Test Suite (NIST STS): We used NIST STS to run a battery of statistical tests against the converted binary data found. NIST STS requires the following input parameters to specify the sequence length, mode of input, tests to run, and number of bitstreams. Most of the parameters required remained the same throughout the testing process. The sequence length remained at the default value of 100,000 bits, and the input was either a converted text or binary file. All tests provided by NIST STS were run on the data with the default value of ten bitstreams.

V. Evaluation and Discussion

Results from the NIST STS tests are shown in Table II. 10 named tests represent the entropy findings. These tests are discussed in Section II-B. All integers represent the proportional result (i.e. 9 is equivalent to a 9/10 in the test output).

Table III ranks the tested entropy sources based on a percentage of proportional results. 10 tests are listed, each with a maximum of 10 in each proportion, so the numerators for each result per test are added and divided by 100 to get the resulting percentage. This is not a percentage of tests passed; rather, it is a percentage representing the overall strength of the entropy.

We interpret entropy strength as a spectrum, as opposed to a binary result regarding the sufficiency of the entropy. Since the only generator and non-static data set we tested was the Real Random EAAS API, we cannot verify the intrinsic characteristics of randomness provided by the other physical sources of entropy by any means other than reviewing existing academic literature. The EAAS Server presents the strongest entropy of the tested sources. It is approximately 3.25 times stronger than the next strongest entropy source, barometric pressure (27%). Humidity and wind speed both have a strength percentage of 21%, so they are both listed as position four. Wind direction and thermal noise are ranked second-to-last and last, with entropy strength percentages of 10% and 0%, respectively.

VI. Conclusion

True randomness is necessary for generating strong cryptographic function components. However, random numbers created by PRNGs is insufficient and can lead to hidden vulnerabilities which are difficult to mitigate caused by non-random seeds used within cryptographic algorithms. Instead, physical sources with intrinsic randomness should be used to generate these seeds. Our contributions include looking at six potential sources of physical randomness, chosen based on the results of an academic literature review, and testing data generated by these sources for the strength of their entropy using the NIST STS. Our results demonstrate that the Real Random EAAS server is the strongest source of entropy of the tested potential entropy sources. The weakest potential entropy source is thermal noise.

The poor outcome produced by testing thermal noise surprised us; the literature review advises that it should be a strong point of entropy. This outcome may be due to the limitations on data collection that we faced. Data set limitations that may have impacted the outcome include the lack of control over how much data the authors generated and the lack of control how the authors presented and formatted that data. The ability to test more data overall, or to being able to feed more data directly from a generator into the NIST, as was done for the EAAS server, could have impacted the outcome.

In the future, generators for wind speed and direction, humidity, barometric pressure, and thermal noise should be created and/or located. The creation of generators for these data types will allow for the study of their intrinsic characteristics and generator layout for randomness property analysis. More thorough testing of their output data may be performed as well, providing more clarity regarding the quality of their entropy. If the creation or location of generators is not possible, one should get multiple data sets of each tested data type and test the data provided by each of those data sets to increase the studied sample size. The time limitations of the project and the challenges presented by using public APIs prevented a larger sample size from being developed for this project.

TABLE II
NIST STS RESULTS

Source	Frequency	Block Frequency	Cumulative Sums	Runs	Longest Run	Rank	FFT	Approximate Entropy	Serial	Linear Complexity
EAAS Server	9	10	9	10	10	10	10	0	10	10
Thermal Noise	0	0	0	0	0	0	0	0	0	0
Wind Direction	0	0	0	0	0	0	0	0	0	10
Wind Speed	0	0	0	0	0	10	1	0	0	10
Humidity	0	0	0	0	0	9	0	0	2	10
Barometric Pressure	2	0	1	1	1	10	3	0	0	9

TABLE III
ENTROPY SOURCE PASS RATE

Position	Source	Percentage
1	EAAS Server	88%
2	Barometric Pressure	27%
4	Humidity	21%
4	Wind Speed	21%
5	Wind Direction	10%
6	Thermal Noise	0%

ACKNOWLEDGMENT

We would like to thank Doug Hill and Real Random, LLC for sponsoring this project and providing access to the EAAS API. We would also like to thank Sumita Mishra for her guidance and support during this project.

REFERENCES

[1] R. Katyal, A. Mishra, and A. Baluni, "True random number generator using fish tank image," *International Journal of Computer Applications*, vol. 78, no. 16, 2013.

[2] O. T. RANDOMNESS, "The importance of true randomness in cryptography," 2017. [Online]. Available: https://www.insidesecure.com/Media/Files/Whitepapers/The-Importance-of-True-Randomness-in-Cryptography

[3] L. Crocetti, S. Di Matteo, P. Nannipieri, L. Fanucci, and S. Saponara, "Design and test of an integrated random number generator with all-digital entropy source," *Entropy*, vol. 24, no. 2, 2022. [Online]. Available: https://www.mdpi.com/1099-4300/24/2/139

[4] H. Corrigan-Gibbs, W. Mu, D. Boneh, and B. Ford, "Ensuring high-quality randomness in cryptographic key generation," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 685–696.

[5] J. Kilgallin and R. Vasko, "Factoring RSA keys in the IoT era," in *2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. IEEE, 2019, pp. 184–189.

[6] A. Vassilev and T. A. Hall, "The importance of entropy to information security," *Computer*, vol. 47, no. 2, pp. 78–81, 2014.

[7] M. Stipčević and Ç. K. Koç, "True random number generators," in *Open Problems in Mathematics and Computational Science*. Springer, 2014, pp. 275–315.

[8] M.-J. O. Saarinen, G. R. Newell, and B. Marshall, "Building a modern TRNG: An entropy source interface for risc-v," in *Proceedings of the 4th ACM Workshop on Attacks and Solutions in Hardware Security*, ser. ASHES'20. New York, NY, USA: Association for Computing Machinery, 2020, p. 93–102. [Online]. Available: https://doi.org/10.1145/3411504.3421212

[9] J. D. Hart, Y. Terashima, A. Uchida, G. B. Baumgartner, T. E. Murphy, and R. Roy, "Recommendations and illustrations for the evaluation of photonic random number generators," *APL Photonics*, vol. 2, no. 9, p. 090901, 2017.

[10] "We believe in building a more secure internet," 2022. [Online]. Available: https://realrandom.co/about-us/

[11] X. Fang, B. Wetzel, J.-M. Merolla, J. M. Dudley, L. Larger, C. Guyeux, and J. M. Bahi, "Noise and chaos contributions in fast random bit sequence generated from broadband optoelectronic entropy sources," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 3, pp. 888–901, 2014.

[12] L. E. Bassham III, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, E. B. Barker, S. D. Leigh, M. Levenson, M. Vangel, D. L. Banks *et al.*, "SP 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications," 2010.

[13] R. G. Brown, D. Eddelbuettel, and D. Bauer, "Dieharder: A random number test suite," 2022. [Online]. Available: https://webhome.phy.duke.edu/~rgb/General/dieharder.php

[14] L. E. Bassham, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, E. B. Barker, S. D. Leigh, M. Levenson, M. Vangel, D. L. Banks, N. A. Heckert, J. F. Dray, and S. Vo, "Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications," Gaithersburg, MD, USA, Tech. Rep., 2010.

[15] B. Valtchanov, V. Fischer, A. Aubert, and F. Bernard, "Characterization of randomness sources in ring oscillator-based true random number generators in FPGAs," in *13th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems*. IEEE, 2010, pp. 48–53.

[16] S. Yasuda, H. Satake, T. Tanamoto, R. Ohba, K. Uchida, and S. Fujita, "Physical random number generator based on MOS structure after soft breakdown," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 8, pp. 1375–1377, 2004.

[17] M.-S. Kim, I.-W. Tcho, and Y.-K. Choi, "Strategy to enhance entropy of random numbers in a wind-driven triboelectric random number generator," *Nano Energy*, vol. 89, p. 106359, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2211285521006145

[18] M.-S. Kim, I.-W. Tcho, S.-J. Park, and Y.-K. Choi, "Random number generator with a chaotic wind-driven triboelectric energy harvester," *Nano Energy*, vol. 78, p. 105275, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2211285520308521

[19] M. P. Pawlowski, A. Jara, and M. Ogorzalek, "Harvesting entropy for random number generation for internet of things constrained devices using on-board sensors," *Sensors*, vol. 15, no. 10, pp. 26 838–26 865, 2015. [Online]. Available: https://www.mdpi.com/1424-8220/15/10/26838

[20] S.-M. Cho, E. Hong, and S.-H. Seo, "Random number generator using sensors for drone," *IEEE Access*, vol. 8, pp. 30 343–30 354, 2020.

[21] J. Bonneau, J. Clark, and S. Goldfeder, "On bitcoin as a public randomness source," Cryptology ePrint Archive, Report 2015/1015, 2015, https://ia.cr/2015/1015.

[22] N. Torii, D. Yamamoto, and T. Matsumoto, "Evaluation of latch-based physical random number generator implementation on 40 nm ASICs," in *Proceedings of the 6th International Workshop on Trustworthy Embedded Devices*, ser. TrustED '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 23–30. [Online]. Available: https://doi.org/10.1145/2995289.2995292

[23] M. Matsumoto, S. Yasuda, R. Ohba, K. Ikegami, T. Tanamoto, and S. Fujita, "1200m2 physical random-number generators based on sin mosfet for secure smart-card application," in *2008 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, 2008, pp. 414–624.

[24] C. R. S. Williams, J. C. Salevan, X. Li, R. Roy, and T. E. Murphy, "Fast physical random number generator using amplified spontaneous emission," *Opt. Express*, vol. 18, no. 23, pp. 23 584–23 597, Nov 2010. [Online]. Available: http://opg.optica.org/oe/abstract.cfm?URI=oe-18-23-23584

[25] A. Elsonbaty, S. F. Hegazy, and S. S. Obayya, "Numerical analysis of ultrafast physical random number generator using dual-channel optical chaos," *Optical Engineering*, vol. 55, no. 9, p. 094105, 2016.

[26] A. T. Erozan, G. Y. Wang, R. Bishnoi, J. Aghassi-Hagmann, and M. B. Tahoori, "A compact low-voltage true random number generator based on inkjet printing technology," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 6, pp. 1485–1495, 2020.