Utilisation of IoT Systems as Entropy Source for Random Number Generation

Oğuzhan ARSLAN[1†] and İsmail KIRBAS[2†]
oguzhan_arslan@outlook.com ismailkirbas@mehmetakif.edu.tr
[†]Mehmet Akif Ersoy University, Institute of Science and Technology, Computer Engineering, Burdur, Turkey

Abstract
Using random numbers to represent uncertainty and unpredictability is essential in many industries. This is crucial in disciplines like computer science, cryptography, and statistics where the use of randomness helps to guarantee the security and dependability of systems and procedures. In computer science, random number generation is used to generate passwords, keys, and other security tokens as well as to add randomness to algorithms and simulations. According to recent research, the hardware random number generators used in billions of Internet of Things devices do not produce enough entropy. This article describes how raw data gathered by IoT system sensors can be used to generate random numbers for cryptography systems and also examines the results of these random numbers. The results obtained have been validated by successfully passing the FIPS 140-1 and NIST 800-22 test suites.

Keywords:
Internet of Things, Cryptography, Random Number Generators, Webcam Sensor, Light Sensor.

1. Introduction

The term "Internet of Things," or IoT in short, is derived from the phrases "object" and "internet" and is one of the subjects that has been the subject of numerous studies in recent years. There are billions of users worldwide who use the global system of connected computer networks known as the Internet. By facilitating the transmission of information between individuals, this global system has become a crucial component of our daily lives. The Internet of Things is the most used terminology, although it has terminological counterparts such as Internet of Everything (IoE), Web of Things (WoT), Web of Everything (WoE), and Machine to Machine (M2M) [1]. The idea of the Internet of Things (IoT) has come to mean a network of interconnected devices that can interact with one another by connecting to the internet without the help of a third party.IoT devices can access cloud-based resources to collect data and extract the collected data, make authorisation arrangements, and make decisions by analysing the collected data with the help of algorithms [2]. Random number generation is critical in many fields because it is used to simulate uncertainty and unpredictability. This is important in fields such as computer science, cryptography, and statistics, where randomness is used to ensure the security and reliability of systems and processes. In computer science, random number generation is used to create randomness in algorithms and simulations, as well as to generate passwords, keys, and other types of security tokens. In cryptography, random numbers are used to generate secure keys for encrypting and decrypting data, as well as to create random challenges in authentication protocols. In statistics, random number generation is used to sample data and to perform statistical tests. Overall, random number generation is a critical component of many systems and processes that rely on uncertainty and unpredictability to function correctly and securely. The raw data needed by random number generators in cryptographic systems can be obtained using the information gathered. This article will describe how these raw data can be utilised to create fixed-length keys that can be incorporated into algorithms that will protect the security of vital communication systems.

Secure communication system architecture, encryption methods, and random number generators (RNG) are the foundations of cryptography. Private keys and secret keys are generated using the distinctive random numbers produced by the RNG. RNGs are divided into two groups: real and pseudo. Because they are simpler to operate, pseudo random number generators are selected more often. Because the quick generation of random numbers without the need for any hardware is a significant cost benefit. On the other hand, true random number generators, which are crucial for secure communication systems, incorporate non-deterministic numbers as a noise source. Expensive gear is needed to capture the genuine unpredictability of the environment. The random numbers that will be produced must exhibit high statistics, be unpredictable, have a consistent structure, and use hardware rather than pseudo RSUs in terms of confidentiality. Some mathematical conditions (randomness tests) must be satisfied if the produced numbers are used in sensitive contexts, such as cryptography systems. The general encryption structure of a text message between a sender and a recipient is depicted in Figure 1.

Manuscript received April 5, 2024
Manuscript revised April 20, 2024
https://doi.org/10.22937/IJCSNS.2024.24.4.9

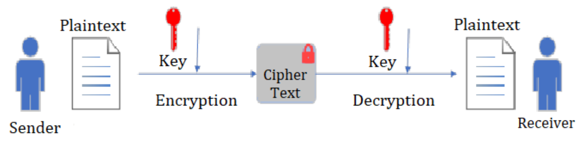

Figure 1. General structure of encryption.

As seen in Figure 1, a text that the sender requests be encrypted is fed into the encryption process with the aid of the key. In this paper, we will talk about how to make the secure key that encryption techniques need.

Previous studies used either internal or external methods to obtain the seed values for random number generators. Post-processing algorithms use input variables like the system clock, mouse movements, CPU data, image or sound data, random functions in programming languages, etc. as seeds. After some time, the numbers generated in this manner begin to repeat and exhibit predictable behaviour. In this project, the values obtained from the Tesla sphere [3], which was used by Nicola Tesla in 1891 to transmit electricity wirelessly as a noise source, will be converted into digital data. After being subjected to a post-processing algorithm with a minicomputer (raspberry pi), fixed-length, unique, unpredictable, and chaos-based number sequences will be obtained. The chaotic environment needed for random number generation is created by gathering information from electrical radiations that are randomly distributed across the sphere, from its centre outward. The input source's chaotic character will guarantee the development of irregular, independent sequences. It is predicted that it will close the knowledge gap in this area and help with the issue of acquiring the seed value of the random number generators used today.

2. Method

To make the secret information between two or more communicating points unintelligible, cryptology, which is a cypher science, encrypts it using a variety of techniques. The secret information is subsequently decrypted on the receiving side. It is a collection of approaches and applications built on high level mathematical ideas [4]. As indicated in Figure 2, the two branches of cryptology are cryptography and cryptanalysis.

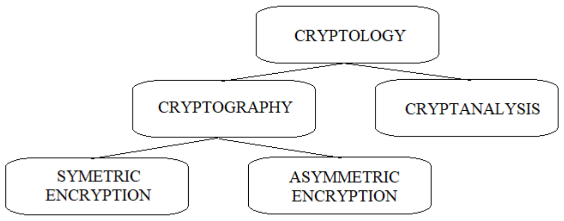

Figure 2. Cryptology.

The phrases "secret" and "writing," which refer to secret writing, are the roots of the word "cryptography." A sender runs the risk of having his communication intercepted and changed when using open networks to convey it to a recipient. Plain text is the message that is in danger here. Encryption is the process of masking a message's content. The plaintext is transformed through this procedure into an encrypted format that is incomprehensible to others. This data could be either encrypted data for storage or a message that is encrypted for transmission. Decryption is the procedure through which the receiving party transforms the cypher text back into plain text [5]. The process of looking for the ciphertext's solution is known as cryptanalysis. Finding potential flaws in cryptographic systems and information breaches is the fundamental goal of cryptanalysis, which is based on exceedingly complex mathematical calculations.

2.1 Encryption Algorithms

Encryption is the process of masking a message's content. The plaintext is transformed through this procedure into an encrypted format that is incomprehensible to others. This data could be either encrypted data for storage or a message that is encrypted for transmission. Decryption is the procedure through which the receiving party transforms the cypher text back into plain text.

In symmetric encryption methods, the encryption algorithm subjects the encrypted message to several procedures before it can be transferred. Figure 3 illustrates how the sender encrypts the message using the encryption key throughout these procedures. Symmetric key encryption techniques use the same keys for encryption and decryption [6]. AES (Advanced Encryption Standard), DES (Data Encryption Standard), and 3DES are popular symmetric encryption techniques today (Triple DES).

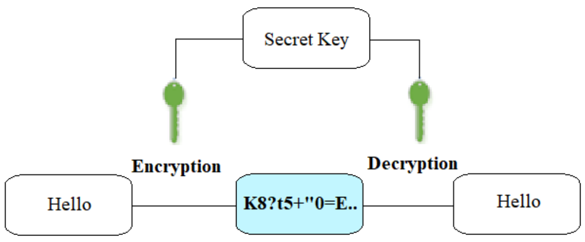

Figure 3. Symmetric encryption and decryption.

Public key encryption is another name for asymmetric encryption methods. For encryption and decryption, there is a public key and a private key. Asymmetric encryption techniques boost the computer's processing capability by using very big prime numbers [5]. Asymmetric cryptography uses public key infrastructure because long keys and lengthy computations are required [7]. The fundamental framework of asymmetric cryptography is depicted in Figure 4.

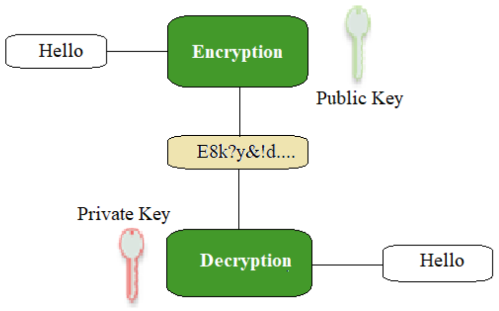

Figure 4. Asymmetric encryption and decryption.

The most popular symmetric encryption technique in use right now is called the Advanced Encryption Standard. AES is a highly effective symmetric key block cipher in terms of both security and performance. The key sizes that can be used for encryption and decryption are 128, 192, and 256 bits [8]. Some of the main characteristics of an encryption algorithm are the following: Confidentiality, integrity, irrefutability, accessibility and identity control [9].

2.2 Random Number Generators

Wherever unpredictability is required, such as in computer games, games of chance, and encryption, random number generators can be utilized. Figure 5 illustrates the division of random number generation into "real" and "pseudo" categories. Pseudo RNGs use algorithms to generate their output, therefore after a while the output data starts to repeat itself on a regular basis. The output data is anticipated to be non-periodic since the source of randomness in a true RNG is based on a chaotic source of uncertainty [10].

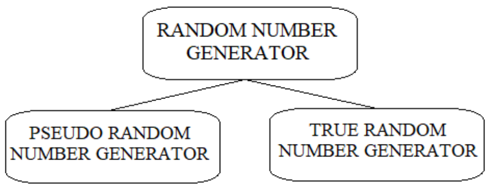

Figure 5. Random number generators.

Systems that produce random numbers deterministically are known as pseudo RNGs. They have benefits over actual random number generators, including ease of creation and an inexpensive cost. By examining its value at any time when the algorithm is compromised, the subsequent outputs can be anticipated [11]. In secure communication systems that demand confidentiality, this prediction may result in significant security issues [12]. True RNGs are systems that employ the chaotic randomness of nature to produce numbers by post-processing with an algorithm. For instance, statistical data gathered by remote monitoring of a plant in an agricultural field or random raw data that cannot be predicted with data obtained from the measurement sensor attached to an animal's foot can be obtained. The numbers exhibiting poor statistical features are post-processed to demonstrate greater statistics after the sampling procedure [13].

Testing for randomness ensures that the post-processed datasets from the entropy source are accurate and realistic. Bit sequences obtained using various sensing sources (camera and light sensor) and methodologies (mode method, last bit extraction, and hash algorithms) will be examined in this research paper's monobit and poker test findings. The ratio of ones to zeros in a sequence is compared in the monobit test. If there are more than 9725 ones in a sequence of 20000 bits, the test is successful. If there are fewer than 9725, the test is unsuccessful [14].

Post processing algorithms will be employed to refine the raw data and boost unpredictability [15]. One of the most popular post-processing techniques, the XOR algorithm, can be characterized as two-bit inputs producing a one bit output. The hash algorithms utilized in this paper are Sha256 and Md5.

3. Experimental Study

The values obtained by using sensors such as temperature, pressure, light, gas, humidity and pH [16], [3], [17] can be used as seed values for random number generators. Ansari et al. created a real random number generator using ldr and sound sensors connected to an Arduino microcomputer [18]. Tuncer and Genç proposed a random number generator based on the GPS sensor in mobile phones and human movement [19]. Yaşar et al. used the random function of the C programming language and the sha256 summarisation algorithm to generate random integers [20]. In his research, Chen obtained random numbers with video and audio noise with a camera [21]. Etem and Kaya created the random number generator for their research without the need for any hardware, using the LCG (Linear Congruential Generator) algorithm with Trivium as the postprocessor [22]. Özkaynak et al., proposed an algorithm that generates random numbers with the pixel values of the photographs obtained from the mobile device camera [23]. By raising the electrical voltage, Nikola Tesla, who was born in 1856 in the Serbian village of Similjan, made it possible to transmit electrical power wirelessly with a low output current density [24]. Raw data from the Tesla sphere in the physical environment will be collected as a noise source using IoT devices or sensors. The obtained values will then be converted into digital data using a Raspberry Pi device, and if necessary, they will be subjected to post processing algorithms to produce fixed length number sequences.

3.1. RGB Colour Sensor

In this section, the raw data from the Tesla sphere utilized as an entropy source that was collected by the TCS34725 colour sensor connected to the Raspberry Pi will be analysed. This sensor additionally measures colour temperature and colour irradiance in addition to colour values. By combining the primary colours of red, green, and blue, colour sensors try to get colour values between 0 and 255. These sensors compare the light from the sensor striking the substance with the light values received by reflecting off the material to arrive at the result. Male-female intermediate cables are used to link the GND, SCL, SDA, and 3V3 pins on the colour sensor to the corresponding pins on the Raspberry Pi device on the breadboard. Figure 6 depicts the overall appearance of the experimental set created with the RGB sensor.

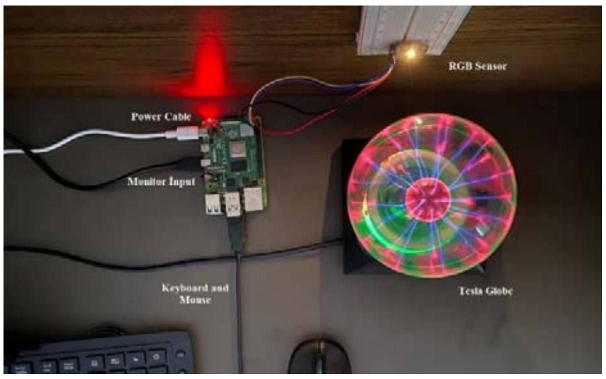

Figure 6. General view of the RGB sensor system.

The following list of components makes up the system, whose schematic representation is shown in Figure 7: Raspberry Pi 4, TCS34725 RGB Colour Sensor, Tesla Sphere, Monitor, Keyboard and mouse.

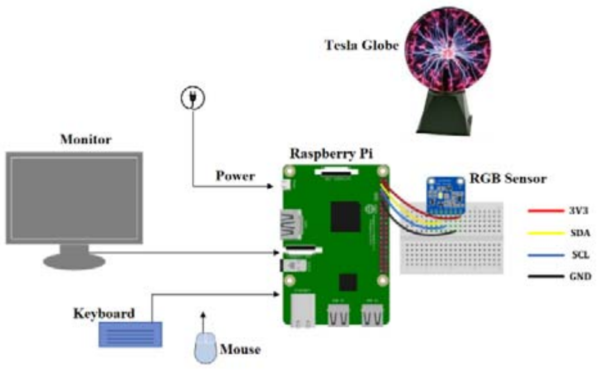

Figure 7. Shape of RGB sensor experiment setup.

A small sphere in the Tesla sphere's centre randomly emits electrical radiations of various colours in the direction of the glass sphere outside. To extract three red, green, and blue values between 0 and 255 from the colour sensor, a Python coding procedure was used. Raw data were gathered from the Tesla sphere in the real world using a colour sensor as a noise source, and the values obtained were then transformed using a Raspberry Pi device into the numerical colour values in Table 3.

Table 3. RGB sensor raw data.

NU.	COLOUR HEXADECİMAL CODE	R-G-B VALUES
1	FFFFFF	(255,255,255)
2	2D2D2D	(45,45,45)
3	FFFFFF	(255,255,255)
4	FFFFFF	(255,255,255)
5	5C5C5C	(92,92,92)
6	5C1010	(92,16,16)
7	5C1010	(92,16,16)
8	5C1010	(92,16,16)

Raw data including RGB values of (45,45,45), colour temperature of 1391.0K, and colour light intensity of 17.566 lux were evaluated. The raw data collected at any given time was noted to be high-quality numbers, but as time went on, the data produced correlated outcomes and the same values overlapped. The (92,16,16) values acquired from the colour sensor were thought to be unsuitable for use as random number generator seeds because they overlap, are not changeable, and have a relationship to one another. As a result, the RGB sensor test results are not listed under the heading "Analysis Results."

3.2. WEBCAM Sensor

In this section, the raw data from the Tesla sphere utilized as an entropy source that was collected by the webcam sensor attached to the Raspberry Pi computer will be analysed. The movements of the electrical radiations in the Tesla sphere were detected using the webcam attached to the Raspberry Pi through a USB port, and raw data with x and y coordinate values were collected. Figure 8 depicts the overall perspective of the experimental setup created using a webcam.

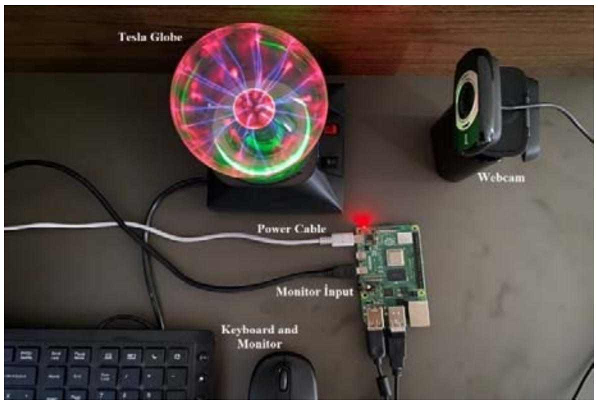

Figure 8. General view of the webcam sensor system.

The following list of components makes up the system, whose schematic representation is shown in Figure 9: Raspberry Pi 4, Webcam, Tesla Globe, Monitor, Keyboard and mouse.

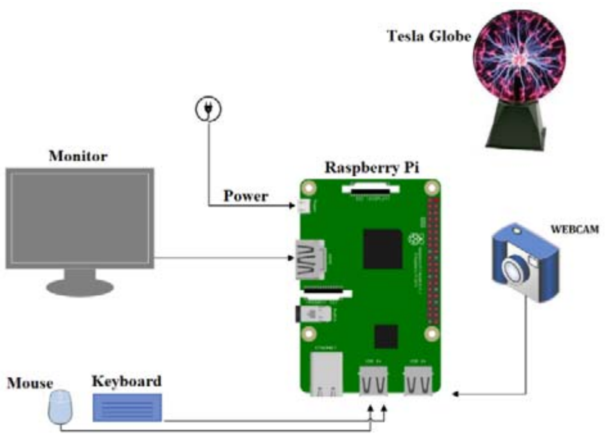

Figure 9. Shape of webcam sensor experiment setup.

The sphere's radiations are identified using OpenCV, a Python computer language package, and the raw data collected from the moving area's x and y coordinates is then examined. Intel introduced OpenCV, an open-source visual library, in 1999. On the Raspberry Pi computer, the necessary installation processes for the OpenCV library, which is utilized in both academic work and commercial applications, were carried out. Following the library's installation, a program in the Python programming language was created that locates moving areas, grids them in, and outputs the weight point's x and y coordinates, as shown in Table 4.

Table 4. Webcam raw data.

Nu.	X Coordinate	Y Coordinate	Elapsed Time (sec)
1	324	305	0.10
2	282	302	0.091
3	197	121	0.078
4	193	82	0.088
5	212	108	0.082
6	260	329	0.078
7	364	133	0.067
8	354	151	0.096

The information gathered in the table above serves as the random number generator's seed values. These variables were used to generate outputs of fixed length using 4 distinct techniques. The first approach entails translating the remainder (Mod 16) into the hexadecimal number system after dividing the x and y coordinate values by 16, respectively. In Table 4, the remainders that were produced after applying the Mod 16 method to the numbers in the second row (282 and 302) correspond to the hexadecimal values "10" and "e," respectively. According to the residual values obtained using this method, the x and y coordinates produced when the webcam sensor detects movement provide an eight-bit output (1010, 1110). Until the specified fixed key length is reached, the motion detection cycle is repeated. The second method involves converting the coordinate values to a binary number system and taking the last bit.

The third-row values (197 and 121) in Table 4 have last bit values of "1" for both coordinate data (after conversion to binary by the last bit method). According to the final bit values discovered using this method, the x and y coordinates formed when the webcam sensor detects movement generate a two-bit long output. Until the specified fixed key length is reached, the motion detection cycle is repeated. The coordinate values are entered into the Md5 and Sha256 hash algorithms to complete the third and fourth methods. After using XOR post processing, the output is obtained by independently summing the x and y coordinate values. These techniques produced 1024-bit outputs, which were then submitted to a monobit randomness test to ensure their randomness. The section under "Analysis Results" will assess the test results.

3.3. Light Sensor

This part will analyse the unprocessed data collected by the LDR sensor attached to the Raspberry Pi from the Tesla sphere used as an entropy source. Utilized by the Raspberry Pi device, the LDR is a sensor that gauges light intensity in proportion to the amount of light that strikes it. The amount of light hitting the LDR will determine how much energy the capacitor receives. The time until logic 1 will provide the light intensity since the Raspberry Pi will identify the capacitor charging as logic 1 when it happens. Male-female intermediate wires on the breadboard are used to link the light sensor and capacitor to the Raspberry Pi device's GND, GPIO3, and 3V3 pins. The light values displayed in Table 5 were collected from the Tesla sphere, which serves as the noise source. Figure 10 shows how the experiment set made with the LDR sensor looks as a whole.

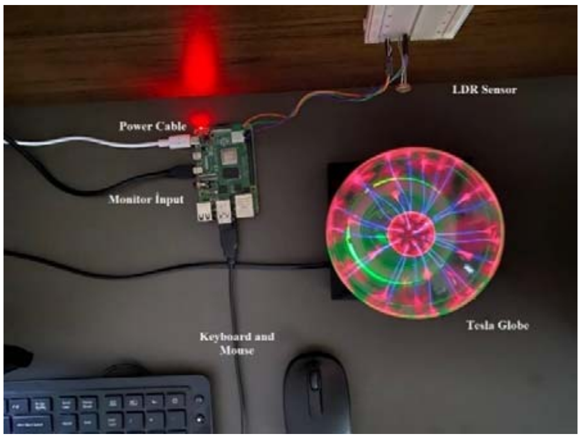

Figure 10. General view of the LDR sensor system.

The following list of components makes up the system, whose schematic representation is shown in Figure 11: Raspberry Pi 4, LDR Sensor, Tesla Globe, Monitor, Keyboard and mouse.

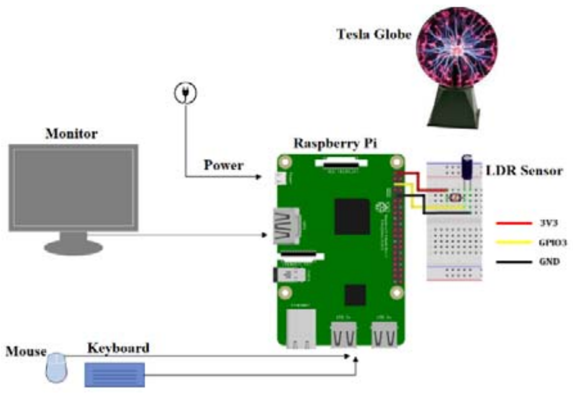

Figure 11. Shape of LDR sensor experiment setup.

Table 5. LDR sensor raw data.

Nu.	Measured Light Intensity	Elapsed Time (sec)
1	1090	0.1021
2	1067	0.1019
3	1098	0.1017
4	1094	0.1019
5	638	0.1023
6	654	0.1023
7	1102	0.1020
8	1061	0.1021

The data obtained in Table 5 serves as the random number generator's seed values. Using these data, four distinct strategies, as described in Section 4.2, were used to produce outputs of fixed length. The first technique is the remainder (Mod 16), which is achieved by dividing the light values by 16. The second approach involves converting the light values to binary and obtaining the final bit. The third and fourth methods are obtained by using the Md5 and Sha256 hash algorithms, respectively. These techniques led to the creation of 1024 bit outputs, similar to those used in the webcam sensor section, which were then subjected to a monobit randomness test in order to verify the unpredictability. The section under "Analysis Results" will assess the test results.

3.4. Comparison of Analysis Results

3.4.1. Monobit Test Analysis Results

The monobit test results from three different sources (Pseudo, Webcam, and LDR Sensor) are compared in this study. The frequency test, sometimes referred to as the monobit test, is discovered by counting the occurrences of the integers 0 and 1 in the sequence. 512-bit values should be one- and 512-bit values should be zero in the 1024-bit long outputs acquired from the sensors in the preceding section. The 1024-bit sequence's monobit test result, which was produced using the Random function in the Python programming language to produce pseudorandom numbers, is shown in Table 6 and shown graphically in Figure 12. The distance between one and zero for the 1024-bit sequence is 34.

Table 6. Monobit test of pseudo random number generation.

Nu.	Expected	Observed
Number of 1's	512	495
Number of 0's	512	529

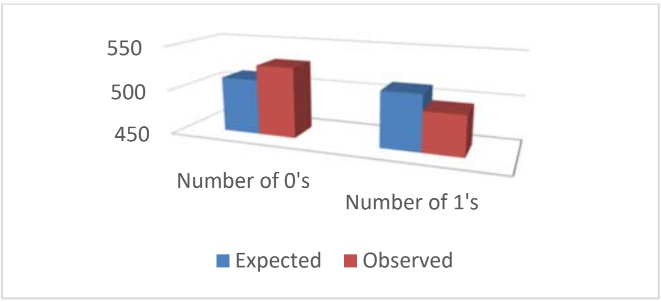

Figure 12. Monobit test chart of pseudo random number generation.

The monobit test results of the output sequences produced by four different techniques using a length of 1024 bits are given in Table 7. The graphical representation is shown in Figure 13, and raw data with x and y coordinate values were obtained by detecting the movements of the electrical radiations in the Tesla sphere using the Webcam sensor. The difference between one and zero for the 1024-

bit sequence using the Mod 16 approach is 10, the Md5 method is 20, the Sha256 method is 18, and the sequence created by omitting the last bits has a difference of 24. The sequence acquired using the Mod 16 approach was found to be the most similar to the expected values, while the sequence obtained using the last bit method was found to be the furthest from them. In this test, it was found that, in comparison to the pseudorandom number produced by the computer, the numbers generated by all techniques employing the Webcam sensor produced good results.

Table 7. Monobit test with webcam.

Nu.	Expected	Observed			
		Mod 16	Md5	Sha256	End Bits
Number of 1's	512	507	522	521	524
Number of 0's	512	517	502	503	500

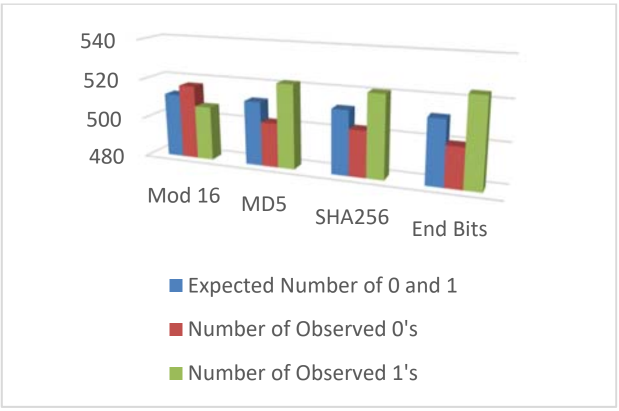

Figure 13. Monobit test chart with webcam.

Table 8 lists the findings of the 1024-bit long arrays' monobit tests, which were conducted using 4 different techniques to gauge the radiation strength in the Tesla sphere using an LDR sensor. Figure 14 shows a graphical representation of the data. When using the Mod 16 method, the difference between one and zero for the 1024-bit array is seen to be 22, when using the Md5 method it is 12 and the Sha256 method to be 32, and when using the array created by eliminating the final few bits it is 16. The sequence acquired using the Md5 method was found to be the most similar to the expected values, while the sequence obtained using the Sha256 approach was found to be the furthest from them. In this experiment, it was found that the numbers generated using any of the LDR sensor's methods performed better than the pseudorandom numbers produced by the computer.

Table 8. Monobit test with LDR sensor.

Nu.	Expected	Observed			
		Mod 16	Md5	Sha256	End Bits
Number of 1's	512	501	506	528	520
Number of 0's	512	523	518	496	504

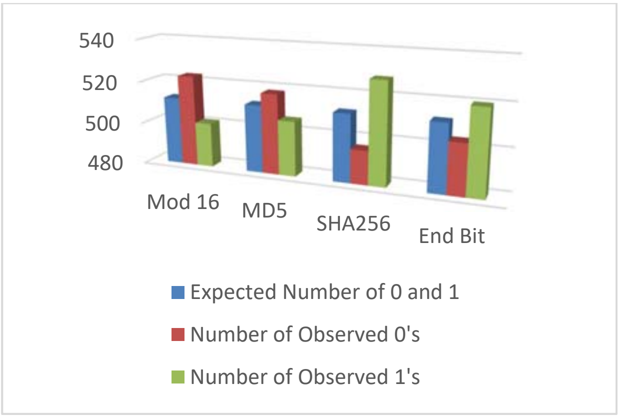

Figure 14. Monobit test chart with LDR sensor.

3.4.2. Poker Test Analysis Results

The poker test results from three different sources (Pseudo, Webcam, and LDR Sensor) are compared in this study. In this test, 5000 numbers are produced by dividing a random sequence of 20000 bits into blocks of four bits. Numbers are expressed in the hexadecimal base using these four bits. The computed poker value must fall between 1.03 and 57.4 in order to pass the test.

Table 9 and Figure 15 show the poker test outcome for the 20000-bit sequence produced by the Random function in the Python computer language, which generates pseudo-random numbers. The poker value for a 20000-bit sequence was found to be 13.9904.

Table 9. Poker test of pseudo random number generation.

Nu.	Expected	Observed
Poker Values (X)	1.03 < X < 57.4	13.9904

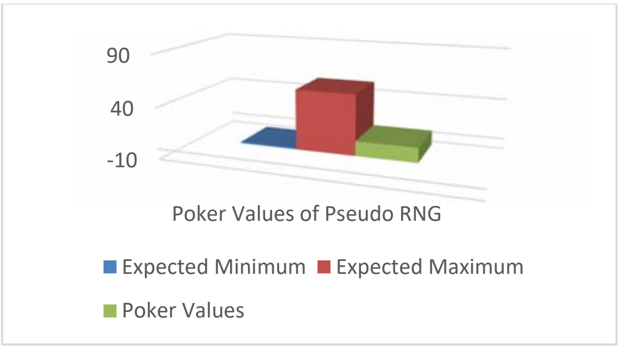

Figure 15. Poker test chart of pseudo random number generation.

After using a webcam sensor to monitor the movement of electrical radiations in the Tesla sphere and obtaining raw data with x and y coordinate values, the results of the poker test for 20000 bit output sequences generated by four different methods are shown in Table 10 and the graphical representation is shown in Figure 16. The poker value of the 20000 bit sequence is 21.4720 for the Mod 16 technique, 0.8256 for the Md5 method, -5.1647 for the Sha256 approach and 10.1504 for the sequence produced by skipping the last bits. It is observed that Mod16 and the last bits method passed the test successfully.

Table 10. Poker test with Webcam.

Nu.	Expected	Observed			
		Mod 16	Md5	Sha256	End Bits
Poker Value (X)	1.03 < X < 57.4	21.472	0.8256	-5.1647	10.1504

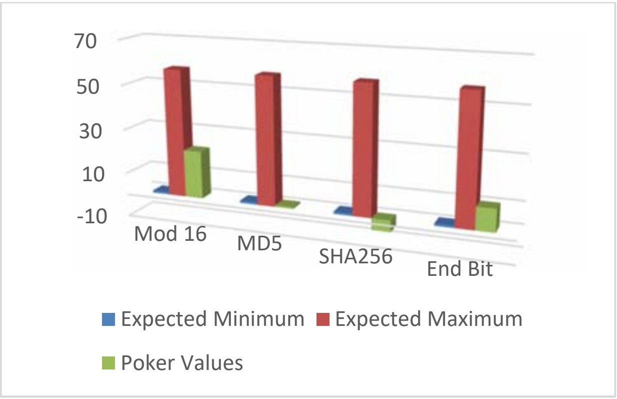

Figure 16. Poker test chart with Webcam.

The poker test results for 20000 bit long sequences obtained using 4 different techniques by measuring the radiation intensity in the Tesla sphere with the LDR sensor are given in Table 11 and the graphical representation is given in Figure 17. The poker value of the 20000 bit sequence is 16.3392 for the Mod 16 technique, -54.4511 for the Md5 method, -5.8944 for the Sha256 approach and 20.3136 for the sequence produced by taking the last bits. It is observed that the last bits and Mod 16 method passed the test successfully.

Table 11. Monobit test with LDR sensor.

Nu.	Expected	Observed			
		Mod 16	Md5	Sha256	End Bits
Poker Value(x)	1.03 < X < 57.4	16.3392	-54.4511	-5.8944	20.3136

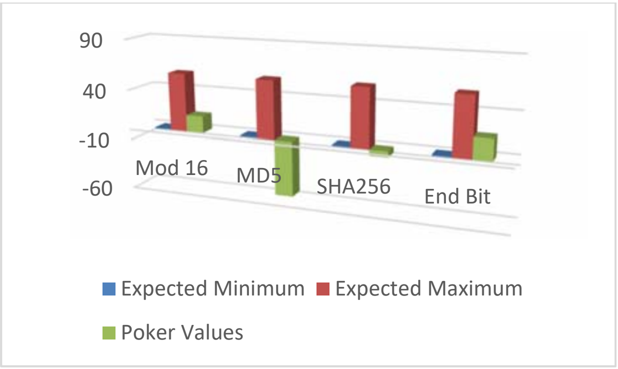

Figure 17. Monobit test chart with LDR sensor.

When the data obtained from RGB, camera and LDR sensors are analysed, it is observed that the same values are obtained in RGB values and close values are obtained as output, although they are not the same values in the LDR sensor. In the camera sensor, two different x and y values were obtained in each cycle, which were not related to each other, and therefore better quality raw data were obtained compared to the other two sensors. When the methods were analysed, it was observed that Md5 and Sha256 did not pass some tests and the Mod 16 method gave better outputs, so the Mod 16 method was used.

3.5. Statistical Test Results

The FIPS 140-1 test suite tests for randomness on a 20000 long generated sequence. This test suite consists of monobit, poker, run and long run tests. The successful results of this test with the camera sensor and Mod 16 method are shown in Table 12.

Table 12. FIPS test results

Test	Expected		Result	
Monobit	9654< X <10346		10037	
Poker	1.03< X <57.4		7.98	
Run	Block Length	Block Number Range	0's Number	1's Number
	1	2267-2733	2494	2520
	2	1079-1421	1247	1191
	3	502-748	662	649
	4	223-402	297	328
	5	90-223	153	150
	6	90-223	148	163
Long Run	<= 34		Passed	

The NIST 800-22 test suite tests randomness on one million long generated sequences. This test suite consists of 16 separate tests, and to be considered successful, the P value produced at each stage of the test must meet the $P \geq 0.01$ requirement. The successful results of this test with the camera sensor and Mod 16 method are shown in Table 13.

Table 13. NIST test results

No.	Test Name	P Value	Result
1	Frequency	0.7634	Successful
2	Block Frequency	0.1559	Successful
3	Run	0.8625	Successful
4	Test for the Longest Run of Ones in a Block	0.7775	Successful
5	Binary Matrix Rank	0.4399	Successful
6	Discrete Fourier Transform	0.6872	Successful
7	Non-Overlapping Template Mathing	0.7312	Successful
8	Overlapping Template Mathing	0.0478	Successful
9	Maurer's Universal Statistical	0.3666	Successful
10	Linear Complexity	0.8760	Successful
11	Serial - 1	0.8307	Successful
12	Serial - 2	0.8601	Successful
13	Approximate Entropy	0.8676	Successful
14	Cumulative Sums	0.9532	Successful
15	Random Excursions (x=+1)	0.3708	Successful
16	Random Excursions Variant (x=-1)	0.6782	Successful

4. Conclusion

In cryptographic applications, randomness is the most crucial element of security and confidentiality. Therefore, the security of the entire system is significantly impacted by the quality of random number generators utilized in various communication contexts. Because these two can be combined, random numbers can be generated as actual, fake, or hybrid. To assess the quality of these generated numbers, several statistical tests are performed. The entropy of the noise source is intimately related to the security of RNGs. By getting the seed value from the physical world using IoT sensors, this study aims to improve entropy levels. The sensors created in the Raspberry Pi environment were used to collect raw data from the Tesla sphere, the source of the noise. Eight different readings were collected using these two sensors, and they were then examined using the Mod 16, Last Bits, Sha256, and Md5 techniques. The raw data obtained with the Mod 16 approach and the camera sensor produced superior results than the other methods, according to the assessments with the Monobit and Poker tests. Sequences obtained by the Mod 16 method successfully passed the FIPS 140-1 and NIST 800-22 test suites, confirming their randomness.

As a result, it has been discovered through this project that seed values for random number generation can be derived from the sensors of IoT systems, which are currently evolving quickly and which we will encounter more frequently in the future in our daily lives. In contrast to the studies in the literature, using the Tesla sphere as the noise source creates a chaotic environment where random, non-repeating data are collected in the next step. It has been shown that the keys that come from the electrical radiations in the centre of the sphere are more accurate than the keys that come from pseudo-random number generators.

Acknowledgments

This study was supported by Burdur Mehmet Akif Ersoy University Scientific Research Projects Commission. Number 0800-YL-21.

References

[1] N. Gözüaçık. "Parent Based Routing Algorithm for RPL Used in IoT Networks." MSc Thesis, Istanbul Technical University, İstanbul, Türkiye (2015).

[2] M. Conti, A. Dehghantanha, K. Franke, and S. Watson. “Internet of Things security and forensics: Challenges and opportunities.” Future Generation Computer Systems 78, (2018): 544–546.

[3] A. I. Sunny, A. Zhao, L. Li, and S. Kanteh Sakiliba. “Low-Cost IoT-Based Sensor System: A Case Study on Harsh Environmental Monitoring.” Sensors 21, no.1 (2021): 214.

[4] Y. Yalman and İ. Ertürk. "The Use of Steganography in Ensuring Personal Information Security." ÜNAK Existence in the Information Age "Opportunities and Threats" Symposium 2, no. 2 (2016): 215.

[5] E. Atar, O. K. Ersoy, and L. Özyılmaz. "Hybrid Data Compression and Optical Cryptography with Steep Matching Search Method." Journal of the Faculty of Engineering and Architecture of Gazi University 32, no. 1 (2017): 139–147.

[6] M. Yılmaz and S. Ballı. "Development of an Intelligent Selection System for the Use of Data Encryption Algorithms." International Journal of Information Security Engineering 2, no. 2 (2016): 18–28.

[7] F. Maqsood, M. Ahmed, M. Mumtaz Ali, and M. Ali Shah. “Cryptography: A Comparative Analysis for Modern Techniques.” International Journal of Advanced Computer Science and Applications 8, no. 6 (2017).

[8] O. G. Abood, S. Guirguis, and S. K. Guirguis. “A Survey on Cryptography Algorithms,” International Journal of Scientific and Research Publications 8, no. 7 (2018).

[9] A. Coşkun and Ü. Ülker. "Development of a Cryptography Algorithm for National Information Security and Reliability Determination Against Letter Frequency Analysis." Journal of Information Technology 6, no. 2 (2013): 31.

[10] V. Tavas. "Random Number Generators Suitable for Integration." PHd Thesis, Istanbul Technical University, İstanbul, Türkiye (2011).

[11] A. Ş. Demirkol. "Adc Based Random Number Generator with Chaotic Oscillator Input." PHd Thesis, Istanbul Technical University, İstanbul, Türkiye (2007).

[12] M. Huang, Z. Chen, Y. Zhang, and H. Guo. “A Phase Fluctuation Based Practical Quantum Random Number Generator Scheme with Delay-Free Structure.” Applied Sciences 10, no. 7 (2020): 2431.

[13] D. Yosunlu and E. Avaroğlu. "Investigation of Post Processing Algorithms." Journal of Computer Science and Technology 1, no. 2 (2020): 66–73.

[14] E. A. Luengo, M. B. L. Cerna, L. J. G. Villalba, D. Hurley-Smith, and J. Hernandez-Castro. “Critical Analysis of Hypothesis Tests in Federal Information Processing Standard (140-2).” Entropy 24, no. 5 (2022): 613.

[15] F. Özkaynak. “Cryptological Random Number Generators." Turkey Informatics Foundation Journal of Science and Engineering 8, no. 2, (2016): 37–45.

[16] Rehman, A. U., Hussain, M., Idress, M., Munawar, A., Attique, M., Anwar, F., and Ahmad, M. “E-cultivation using the IoT with Adafruit cloud." International Journal of Advance and applied Sciences 7, no. 9 (2020): 75–82.

[17] H. Üçgün, F. Gömbeci, U. Yüzgeç, and N. Yalçin. "Real-time Indoor Air Quality Monitoring System with IoT Based Platform." Bilecik Şeyh Edebali University Journal of Science and Technology 7, no. 1 (2020): 370–381.

[18] U. Ansari, A. K. Chaudhary, and S. Verma. “True Random Number Generator (TRNG) Using Sensors for Low Cost IoT Applications.” 2022 International Conference on Communication, Computing and Internet of Things (IC3IoT), (2022): 1-6.

[19] Y. Genç and S. Arslan Tuncer, "Human Movements Based True Random Number Generation." Bitlis Eren University Journal of Science and Technology 8, no. 1 (2019): 261–269.

[20] S. N. Yaşar, F. Ceren Dikici, E. Tanyildizi, and E. Karaköse. "Design of a Generator Based on Middle Square and SHA3 Algorithm for Randomisation Requirements in Science and Engineering Studies." Fırat University Journal of Science and Technology 33, no. 1 (2021): 81–91.

[21] I. Te Chen. “Random numbers generated from audio and video sources.” Mathematical problems in engineering. (2013).

[22] T. Etem and T. Kaya. "Trivium-Linear Conjugate Generator Based Bit Generation for Image Encryption." Fırat University Journal of Engineering Science 32, no. 1, (2020): 287–294.

[23] F. Ozkaynak, H. I. Ozdemir, and A. B. Ozer. “Cryptographic random number generator for mobile devices.” 2015 23rd Signal Processing Communication Application Conference, (2015): 1733–1736.

[24] Z. E. Sezgin. "Tesla Coil." Maltepe University, (2021).

OĞUZHAN ARSLAN He received his bachelor's degree in computer engineering from Fırat University in 2014 and his master's degree without thesis in information systems from Gazi University in 2021. He continues his master's degree with thesis in the field of cryptography at Burdur Mehmet Akif Ersoy University, Department of Computer Engineering.

İSMAİL KIRBAŞ received his bachelor's degree in electronics and computer education from Kocaeli University in 2000. He received his master's degree from the same university in 2009. He completed his doctoral thesis at Sakarya University, Institute of Science and Technology in 2013. He was given the rank of associate professor in 2018, and since then, he has been focusing on the internet of things, time series analysis, machine learning, mobile and the web-based application development.