# Quantum Random Number Generators and Their Effect on Cryptography

**Nikhil Mallela**

*Received August 20, 2024*
*Accepted January 10, 2025*
*Electronic access January 31, 2025*

With the advent of quantum computing, there is a new potential for significant advancement in random number generation, a critical component in various fields, specifically, cryptography. This paper is a comprehensive literature review which compares classical random number generators (RNGs) with quantum random number generators (QRNGs). Classical RNGs often rely on physical phenomena to develop seeds of true randomness, however this practice is limited by speed, and makes testing difficult as results are impossible to replicate. Furthermore, the security of physical phenomena is based on the phenomena being too difficult to model at the moment, so their security could be compromised with future advancements. This necessitates the use of pseudorandom number generators (PRNGs) to produce usable random numbers, which are based on algorithms, however they are still susceptible to cryptographic attacks. On the other hand, QRNGs exploit the inherent unpredictability of quantum mechanics to generate truly random numbers on a speedy basis, thus promising to resolve many problems with classical generation. This review synthesizes key texts to analyze both methodologies. The findings highlight that QRNGs offer superior randomness and therefore security, and that they offer to revolutionize many industry practices by providing more robust encryption keys. However, limitations in implementation mean that practically, they should only be used in high security situations, and for many lower level applications the use of entropy seeds with PRNGs should be perfectly sufficient.

**Keywords:** Cybersecurity, Randomness, Entropy, Random number generators, Psuedorandom number generators, Quantum computing, Quantum random number generators

## Introduction

The importance of random numbers cannot be overstated. From gambling to scientific analysis, there is a need for random numbers everywhere. One field with special need of them is that of cryptography. Any attempt to offer privacy with computers fundamentally relies on random numbers. The most notable instance of this is encryption, in which a key is required for users to interact with each other securely. The safest way to pick this key is by using a randomly generated number that is random enough that it would be difficult for any malicious hacker to guess. Many classical RNGs face problems such as speed and predictability, as most numbers in use are not truly random. Classical RNGs rely on physical phenomena to develop "seeds" (small, random values capable of being lengthened) of true randomness.

However, this technique is not capable of mass-producing random numbers, so algorithms, i.e. PRNGs, take these seeds and perform a variety of calculations on them to stretch these seeds into longer, even though less random, but importantly more easily manufactured sets of random numbers. This process can have vulnerabilities due to bias, environmental interference, and the deterministic nature of PRNGs. Despite their widespread use, these methods have limitations that compromise their effectiveness. However, with the rise in quantum computing, many seek to build QRNGs that promise to leverage the inherent randomness of quantum mechanics, to offer truly random numbers to use [1].

QRNGs use the unpredictability of quantum phenomena, such as the behavior of photons, to produce random numbers that are free from the biases that plague classical methodologies. This new approach has the potential to revolutionize the generation of random numbers, providing a level of security that classical methods can't achieve.

All this raises several essential questions: Why is classical cryptography not enough in the new era of quantum computing? Which aspects of it shall break, or which aspects already have? The main problem lies in the deterministic processes underlying classical RNGs and their susceptibility to bias and predictability, which cannot keep up with the newfound speed and power of quantum decryption. As quantum computing continues to advance, classical cryptographic methods are increasingly at risk. Popular algorithms like RSA, which rely on the difficulty of factoring large numbers [2],are expected to become vulnerable to quantum attacks due to Shor's algorithm, which shows the future potential of quantum algorithms in quickly factoring

prime numbers[3]. This study seeks to explore the limitations of classical RNGs, the rise and use cases of QRNGs, and how the latter can address novel cryptographic vulnerabilities in the new era.

This paper will be a literature review evaluating the states of both classical and quantum RNGs, detailing the drawbacks and benefits of both, with special attention to the impact recent developments in the field could have on cryptography. It is structured as follows: section  lays out the primary materials and methods used for this literature review and justification for putting these papers in conversation with each other. Section will then offer relevant background on fundamental concepts in classical computing while  will outline the entire process of classical random number generations. Similarly, section  will first explain key concepts in quantum computing that are essential in understanding section  that outlines the entire process of quantum random number generation. After sufficiently explaining both methodologies, section  will fully outline the drawbacks and advantages and drawbacks of both techniques, while section  will generally venture at the impact QRNG will have on cryptography, including their practicality and in which scenarios they would be suited for.

## Methodology

For this research paper, I searched for relevant academic papers. The goal was to gather comprehensive sources on both classical and quantum RNGs in order to analyze their advantages and limitations, specifically with respect to cryptography. The most important qualifying factor in selecting papers was their relevancy, with respect to their number of citations and how recent they were. This section will further detail the synthesis process and rationale in the selection of papers.A PRISMA diagram is offered below in figure 1 to visualize this process.

### Search strategy

To find relevant literature a variety of databases and journals were scanned. Relevant keywords included: RNGs, PRNGs, CSPRNGs, QRNGs, RNG speed, PRNG speed, QRNG speed, entropy sources, and RNG statistical testing. Important qualities in more heavily used papers were the number of citations and recency. I conducted in-depth research on these papers to ensure a thorough understanding of both classical and quantum RNGs. These searches were supplemented by more specific searches relating to statistical tests and the more technical aspects of QRNGs. Papers were then selected based on the depth of technical detail and relevance to contemporary developments. The objective in this was to include foundational texts and recent changes in the field to offer a comprehensive review of the practical applications and theoretical underpinnings of each methodology.

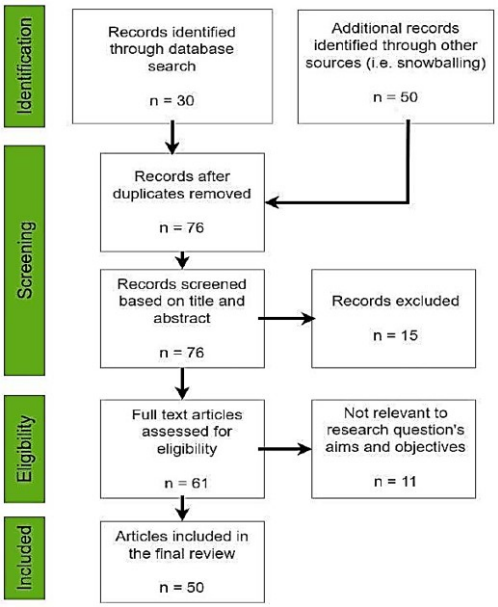


**Fig. 1** PRISMA diagram

### Inclusion criteria

The primary concern in selecting papers was how recent they were, as the field of QRNGs is rapidly developing, so assessments concerning speed and entropy on effectiveness would have to use up-to-date generators. For tables and data, more recent papers were highly prioritized. In introducing the basics of random number generation and quantum mechanics, not as much weight was given to recency, as those fundamentals do not change with time as much. Another important facet which was paid attention to was how many citations a paper had, and papers with more citations were given greater weightage.

### Key Papers' argumentation

The selected papers were organized in such a way to create a comprehensive narrative that would contrast classical and quantum RNGs, by highlighting their strengths and weaknesses, and discussing their implications on cryptography.

**Classical RNGs:** The primary source for this section was the third chapter from the "Handbook of Computational Statistics" by Pierre L'Ecuyer[4], this provides a comprehensive background on classical random number generation. This chapter dives further into the statistical techniques used to test randomness and is instrumental in contrasting random and quantum RNGs. Additional papers were reviewed to explore entropy sources and PRNGs. These papers also discuss the challenges of classical RNGs such as environmental interference and bias that are critical in understanding the need for QRNGs in certain cases. Chief among these papers is "A Simple Unpredictable Pseudo-Random Generator" by L. Blum et. al[5]. which specifically goes

into how PRNGs algorithmically develop the seeds from entropy sources into industry ready products.

**Quantum RNGs:** The main paper concerning QRNGs that was used is "A comprehensive review of Quantum Random Number Generators: Concepts, Classification and the Origin of Randomness" by Vaisakh Mannalatha et. al[6]. This paper offers a full outline of the quantum process in generating random numbers in well-articulated terms. Further papers were used to outline the more technical aspects, such as "Quantum Random Number Generators" by Miguel Herrero-Collantes et. al[7]. and "Quantum Generators of Random Numbers" by Marcin M. Jacak et. al[8]. Respectively, the first paper goes into what qualifies true randomness, and the second bridges the gap between theory and practicality by discussing the commercial uses of QRNGs. Together, they help paint a clear picture of the state of quantum random number generation.

### Synthesis method

To effectively compare and contrast classical an quantum RNGs, the papers were placed in a dialogue with each other to ensure full understanding on both fields of generation. The classical RNG papers provide a foundational understanding on the concept of randomness and how it is tested. This sets a baseline to understand the more advanced quantum techniques. The QRNG papers are placed after the classical ones to highlight technical advancements and some advantages of quantum generation such as true randomness and lack of biases. Papers discussing commercial viability will help in outlining the practicality of QRNGs and assessing the feasibility in adopting QRNGs, and whether they should replace classical RNGs or augment them.

## Classical Cryptography

Since the effectiveness of randomly generated numbers directly depends on how random they are, it is crucial to quantify this "randomness". Two common indicators are uniformity and independence. Another is entropy, which plays a vital role in quantifying randomness by measuring the disorder in a system[6], providing a mathematical technique to compare the randomness of sequences. This concept was applied to cybersecurity by American scientist Claude Shannon, who devised the Shannon entropy formula[9] to quantify the uncertainty associated with a variable. This formula for entropy, $H(x)$, is often given by:

$$H(x) = -\sum p(x)\log(p(x)), \tag{1}$$

where, in simple terms, $p(x)$ refers to the probability of an event occurring and $\log(p(x))$ refers to the "information content" of that event, and the based is assumed as 2. For example, if we flipped a coin, and 1 referred to heads and 2 referred to tails, and we wanted heads, the probability of getting heads, or $p(1)$, would be $\frac{1}{2}$. Then, the log expression would refer to information associated with $x$. So, $\log(p(x))$ would be $\log(\frac{1}{2})$, or $-1$, so we could determine that getting a heads provides 1 bit of information, which is what we would expect as in both binary and coin flips, there are only two options.

Considering the product, events with higher probability would mean less overall entropy, as if we expect the event more then it is less random. The summation then covers all possible events of our variable, while the negative sign ensures our result is non-negative as we take a measure of uncertainty. This formula, in addition to other tests and qualities are then used in measuring the quality of randomness in a set of generated numbers, thus giving us a mathematical technique to compare results.

### Classical RNGs

The process of generating random numbers is kick-started with a "seed" value. A common method in generating this seed is through manual user input.

While this is by far the simplest, it heavily relies on the ability of the user to be unpredictable. Another approach involves using the current time of the system, which constantly changes.

However, this method is predictable if an attacker knows the time of generation. The most robust methods for generating these seeds involve physical phenomena which are currently too difficult to model[10].

Examples of these phenomena include but are not limited to: keyboard input timings, thermal noise, mouse movements, and the timings between atomic decay. These sources are far more unpredictable and thus are far more secure.

After this, a sequence of numbers is generated by applying a variety of calculations to the seed. These numbers usually range between 0 and 1, and show traits similar to truly random numbers. This sequence has uniform distribution, so each number in the specified range has an equal chance of being produced. Once the raw sequence is obtained, they are transformed in order to follow statistical distributions, like exponential decay or bell-shaped functions[11].

This transformation lets us simulate different types of random data for further analysis and modeling. It is not enough that the sequence just looks random or chaotic, as there are often patterns that are not visible by just looking at the sequence[4]. All sequences meet specific criteria to be considered random, such as: all generated numbers needing to be independent of one another, being spread out evenly over a range (usually between 0 and 1) to ensure equal chance of reproduction, and if the device produces in binary, each bit should have an equal chance of appearing.

Statistical testing is crucial to determine the strength of sequences as bad numbers would have disastrous effects in fields such as security or gambling. Tests such as the NIST suite[12], which uses a variety of tests to assess randomness of sequences

by checking for uniformity, unpredictability, and independence. Another example is the Die-Hard test which checks for traits such as frequency and correlation to check for patterns or biases which would show the sequence is not random[6].

This is an incredibly effective process, however it can be a relatively slow and cumbersome process to mass produce random numbers this way. Table 2, simplified from the paper of Pareschi et. al,[13] is offered to show the speed of a variety of classical RNGs:

| RNG | Output bit rate (after post-proc) [Mbit/s] |
|---|---|
| Petrie and Connelly[14] | 1.4 |
| Kinniment and Chester[15] | 4 |
| Bucci et. al.[16] | 10 |
| Bucci and Luzzi.[17] | 1.74 |
| Tokunaga et. al.[18] | 0.04 |
| Holleman et. al.[19] | 0.005 |
| Holleman et. al.[19] | 0.05 |

To improve that aspect of generation, PRNGs step in by taking a short, truly random seed value and applying a deterministic algorithm to it to produce a long, *pseudo*random sequence[10]. While these sequences are not truly random, they are statistically strong and suitable for a variety of practical purposes[5]. The process of true random number generation is still useful in generating initial seed values. The PRNGs, essentially, "stretch" this random seed into a pseudorandom sequence, therefore, the algorithm can be viewed as an "extensor of randomness"[20]. Table 3 from the paper of K. Sathya et. al.[20] is offered to show the remarkable speed of PRNGs:

| PRNG | Speed (Mbps) |
|---|---|
| Francois and Defour[21] | 80,000 |
| Bakiri et. al.[22] | 6950 |
| Barani et. al.[23] | 2.9 |
| Huang et. al.[24] | 21.5054 |
| Huang et. al.[25] | 3.8955 |
| Murillo-Escobar et. al.[26] | 1.7 |
| Lv et. al.[27] | 138.0864 |
| Wang et. al.[28] | 164.3064 |
| Ma et. al.[29] | 449.236 |
| Zhao et. al.[30] | 0.5017 |
| Ozcanhan et. al.[31] | 0.01356 |
| Yu et. al.[32] | 62.5 |
| Mandal et. al.[33] | 408 |
| Alhadawi et. al.[34] | 14.48 |
| Alawida et. al.[35] | 2.1105 |
| Chen et. al.[36] | 598.1 |
| Liu et. al.[37] | 367.88 |
| Lambic et. al.[38] | 34.18 |
| Riaz et. al.[39] | 2.1026 |
| Akhshani et. al.[40] | 873.05 |
| Kalanadhabhatta et. al.[41] | 832 |

This hybrid approach gleans the randomness of physical methods while slightly compromising to achieve the speed of algorithmic methods, thus producing high-quality sequences for many applications across industrial use.

## Quantum Cryptography

The quality of randomness from QRNGs is fundamentally founded upon the inherent randomness in quantum mechanics[42]. There are several concepts in quantum mechanics that are critical in understanding the additional security given by QRNGs. Chief among these properties is that of entanglement, which is a phenomenon where two or more particles are connected in a manner where the state of one particle is dependent on the other[28]. A helpful formula when considering entanglement is the Dirac equation[43]. The equation relates how particles like electrons behave when moving near the speed of light. Electrons have a property called "spin" which describes the orientation of a particle in a certain manner. When two particles become entangled, their particles connect in such a way that measuring the spin of one immediately delivers information on the other, regardless of distance. This property is foundational to quantum cryptography and offers many advantages. While information is usually sent in a manner where each individual bit can be observed and intercepted, entangled particles have a joint state[7]. In this state, any measurement made on one instantly affects the other, despite distance. This property gives extremely secure channels, as any attempt to observe the system will disturb it and reveal an attacker. This is used in a variety of computational protocols such as "quantum key distribution" (QKD)[44], where entangled pairs are used to generate keys.

If an adversary attempts to intrude upon the system, entanglement is disturbed and their presence is detected. Another similar property is non-locality, which is similar to entanglement, but emphasizes the instantaneous nature of entangled particles[6].

### QRNGs

The process of quantum random number generation is similar to classical methodologies in generating a seed through physical phenomena, but differs in which exact phenomena it utilizes. While setting up, subatomic particles like photons or electrons are transformed to create a state necessary for generation. This system displays necessary properties such as superposition and entanglement.

Superposition is a cornerstone of quantum mechanics which allows a qubit, the basic unit of quantum information that is similar to a classical bit, to exist in multiple states simultaneously as it can represent both a 0 and 1 at the same time. A helpful equation to consider is the superposition principle[45], which is

often given by:

$$|\psi_i\rangle = \sum_j C_j |\phi_j\rangle, \tag{2}$$

where $|\phi_j\rangle$ refers to the individual possible positions a particle can be in, and $C_j$ describes the "weightage" each state contributes. The summation is the linear combination of all these states, as all positions exist in a mixture, which is conveyed by $|\psi_i\rangle$. This superposition is essentially, a canvas of probabilities, which is inherently random. When these qubits are entangled, an extra layer of security is added as this makes sure the particles are protected from external influence, and any attempt at tampering with them is detectable. This initial state ensures that the system is ready to generate the highest quality of random numbers.

The measurement of this quantum state is the biggest difference when compared to classical generation. When a system is directly observed, superposition collapses into a typical state, which is usually represented using a basis, a set of vectors used to show a quantum state. In the computational basis, which is the most commonly used one, measurements provide either $|0\rangle$ or $|1\rangle$, similar to classical bits being either 0 or 1. Another common basis is the Hadamard basis[8], where $|0\rangle$ and $|1\rangle$ correspond to $|+\rangle$ and $|-\rangle$. This is used for specific operations and transformations[7], which are easier to do in this representation. This act of measurement is truly unpredictable and random, as the quantum superposition collapses into 1 state.

These raw bits then undergo a variety of post-processing techniques to eliminate biases and increase randomness. These systems can have some errors due to flaws like imperfections in the hardware. Many codes can be used such as the Hamming or Reed-Solomon to correct these errors, while retaining the random nature of the sequence. Another technique is privacy amplification, where residual information is removed which could have been exploited. Other techniques such as hashing are used to whittle the sequence into a smaller set which is more uniform.

The highest quality bits are then extracted from the sequence through methods such as the Von Neumann extractor, Toeplitz hashing, and SHA-256[8]. Taken together, they "purify" the randomness of the sequence, to meet standards.

Finally, the sequence undergoes a verification process, similar to sequences produced traditionally, with a variety of tests such as the aforementioned NIST suite and the Die-Hard test. This process confirms that the sequence is suitable for its myriad applications, ensuring its necessary security properties.

## Comparison

Classical RNGs and PRNGs have several advantages that make them useful in certain cases. The main benefit of using PRNGs is their speed. PRNGs, due to their algorithmic nature, can generate large sets of numbers at a pace incomparable to other types of RNGs. The simplicity of classical RNGs is also important to note. They are often implemented on standard hardware without relatively specialized equipment when compared to QRNGs. Therefore, they are far more accessible and integrable in various systems. Their relative ease also makes them cost-effective as one can avoid needing advanced hardware and very specialized workers.

The deterministic nature of PRNGs is helpful in many cases.

When testing and debugging, it is very important to reproduce certain sequences to consistently find and fix issues[46]. This is because in debugging, programmers need to recreate the exact same conditions repeatedly to identify the source of an issue and see what effect specific changes on the program will have. Similarly, programmers rely on replication in vulnerability testing as the same environment allows researchers to assess what effects potential exploits could have. As similar conditions can be maintained, development and maintenance updates are easier to test. Without a static environment, tests would be both inconsistent and unreliable.

However, this algorithmic nature can also be a disadvantage. If the seed value is known or can be predicted, then the sequence can be reproduced and the product loses its randomness. This can have disastrous implications in fields that rely on the randomness of these numbers, especially cryptography. An attacker being able to predict the random numbers would compromise the keys and security of many systems, thus being able to gain access to important data. To solve these problems, cryptographically secure pseudorandom number generators (CSPRNGs) are often employed[47]. CSPRNGs increase the security of PRNGs by relying on a variety algorithms that make it more difficult to predict or reproduce a sequence without direct knowledge of its internal state. Common methods such as Fortuna and the Yarrow algorithm[48] integrate more entropy sources to strengthen their randomness and reduce vulnerability to attacks. While these countermeasures have been effective in some practices, their implementation must still be carefully managed, as poor entropy sources or flawed cryptographic algorithms can lead to significant vulnerabilities. The quality of randomness in PRNGs is doubtless important to take note of. In high-security cases, they can be liabilities if not implemented perfectly, so other options work far better in these scenarios.

Classical RNGs, specifically those based on physical phenomena, while possessing high levels of entropy, are lacking in speed[49]. This makes them burdensome to use when mass-producing, which is why they are more often used to generate seed values, and then they can be combined with PRNGs. Furthermore, the security of this process is tantamount on the physical phenomena used to generate seed values being difficult to model[6]. Future advancements in being able to model these phenomena, such as thermal noise, could render many of these

secure systems defenseless.

In contrast, QRNGs leverage quantum mechanics to give true randomness, which is vital in cryptography. This true randomness offers a higher level of security than that offered by PRNGs, and they are inherently random so they cannot be modeled to reproduce the same result repeatedly. This benefit can be seen in a variety of attacks. For example, brute force becomes far more costly, as attackers cannot guess patterns in sequences as they would be able to with PRNGs because the randomness of QRNGs has no exploitable patterns. Furthermore, a sequence would be incredibly difficult to reverse-engineer, as there is no pattern or discoverable seed to speak of that could be reverse-engineered or predicted. While QRNGs are faster than classical RNGs, they do not match the speed of PRNGs, especially in situations where quantum infrastructure is hard to implement or not readily available. This makes classical RNGs more practical for certain cases where speed is prioritized over absolute randomness. However, QRNGs strike a balance between speed and randomness that is ideal for cybersecurity applications, particularly in scenarios requiring high entropy. Table 4, simplified from the paper of Mannalatha et. al,[6] is offered to illustrate the speed of a variety of commercially available QRNGs:

| QRNG | Speed (Mbps) |
|---|---|
| IDQ Quantis-PCIe-240M | 240 |
| PicoQuant PQRNG150 | 150 |
| QuatumCTek QRNG100E | 600 |
| ComScire CS128M | 128 |
| Quitessence Labs qstream200 | 1000 |
| Quantum eMotion QNG2 | 1000 |
| EYL QRNG-H | 1000 |
| Qutools quRNG | 50 |
| MPD QRN-128 | 128 |
| Quside PCLe One | 2000 |
| QNU TROPOS QNL-QRNG-X100 | 100 |

In gaming applications such as lotteries and online casinos, speed is a necessity to handle large volumes of players and real-time responsiveness. In this case, speed would be prioritized over strict randomness, so PRNGs would work perfectly. However, in situations like banking or government and military matters, randomness is critical as these are high-security situations, and QRNGs would be well-suited to work here. Despite their advantages, reliance on quantum technology can be a limiting factor, as using QRNGs in environments lacking quantum-compatible technology can prevent their practical adoption. A more comprehensive analysis of the challenges faced in implementation versus their security benefits can provide a clearer understanding of their applicability across a variety of fields. They are also faster than classically hardware-based RNGs, striking a balance between speed and randomness that other RNGs can't[6]. While they don't strictly match the speed of PRNGs, they are still fast enough for most cybersecurity applications.

QRNGs are also often implemented with self-testing protocols to verify the quality of their numbers which provide an additional layer of security that make them useful in situations where utmost security is necessary.

QRNGs also have a few drawbacks which deserve consideration as well. One of their main challenges is that they are both complex and costly in their implementation.

They require highly specialized workers and equipment such as single-photon detectors and accurate measurement systems. Maintenance is also no easy feat, and they require highly controlled environments to use securely. Their lack of pace when compared to PRNGs can also limit their practical use in many lower-security situations which also require high throughput of data. Finally, they are also a new technology when compared to classical RNGs, so they are not as widespread or well understood. The lack of tools and bases for them make QRNGs difficult to integrate for many, which limit their practical use.

### Implications

QRNGs are capable of addressing several problems which plague classical RNGs. Their unique ability to have greater speed while not compromising on randomness make them promising for use across many environments, specifically high-security ones. The physical requirements of QRNGs are practical necessities in important scenarios. In environments such as government data, military communication, and financial transactors, security is paramount and thus QRNGs would be a perfect fit as they do not compromise on the security of classical hardware-based RNGs but are still speedy. The robustness of quantum generated random numbers makes it difficult for many threat actors to find patters or predict output, thus heightening security. Due to their complex and expensive nature, it is in situations such as these that their costs can be justified. It is inevitable that adversaries will respond to the heightened security of QRNGs. As more systems integrate QRNGs, they will develop new methods to overcome these challengers. However, remaining with classical methods could put many systems behind the curve, considering the possibility of currently complex physical phenomena becoming easier to model. PRNGs combined with seeds from classical RNGs will be sufficient for many lower-security situations, but higher-security cases will rely more often on QRNGs due to their unique promises[6]. This change will inevitably lead to further research in this technology, potentially even making these systems cheaper and more scalable in implementation.

## Conclusion

This paper has thoroughly compared classical RNGs and QRNGs, by representing their mechanics in generation, advantages, and limitations. The combination of classical RNGs

with PRNGs are used widely due to their speed and efficacy, however, relying on physical phenomena and decreased levels of randomness runs the risk of significant security problems. Meanwhile, QRNGs utilize the chaos of quantum mechanics to offer a heightened form of randomness with greater speed.

The analysis shows that QRNGs aren't full ready for all scenarios, due to their complexity and cost of implementation, however, they have the capacity to become incredibly useful in high-security situations. As research in this field develops, it is anticipated that they will become cheaper and easier to implement, thus transforming the cybersecurity scene. In conclusion, integrating QRNGs into high-security systems would provide unparalleled security, however, using classical RNGs to develop seeds and PRNGs to lengthen them still remains an effective solution for many lower-level applications. Future research will transform the practicality of QRNGs, thus rendering them as a key piece of secure communications in the days to come.

## Acknowledgments

The author expresses gratitude to Konstantinos Brazitikos for his constant aid and providing pertinent resources, and Melissa Riley and Dr. Aaron Nuñez for their continued support.

## References

1 Ananthaswamy, *How to Turn a Quantum Computer Into the Ultimate Randomness Generator*, https://www.quantamagazine.org/how-to-turn-a-quantum-computer-into-the-ultimate-randomness-generator-20190619.

2 E. Milanov, *The RSA algorithm*.

3 U. Ugwuishiwu, C. Ugwu and C. Asogwa, *An overview of quantum cryptography and shor's algorithm*.

4 P., *L'Ecuyer, Random number generation*, Springer.

5 L. Blum, M. Blum and M. Shub, *A simple unpredictable pseudo-random number generator*.

6 V. Mannalatha, S. Mishra and A. Pathak, *A comprehensive review of quantum random number generators: Concepts, classification and the origin of randomness*.

7 M. Herrero-Collantes and J. Garcia-Escartin, *Quantum random number generators*.

8 M. Jacak, P. Jóźwiak, J. Niemczuk and J. Jacak, *Quantum generators of random numbers*.

9 E. Shannon, *A mathematical theory of communication*.

10 J. Katz and Y. Lindell, *Introduction to modern cryptography: principles and protocols*.

11 G. Marsaglia, A. Zaman and W. Tsang, *Toward a universal random number generator*.

12 L. Bassham, A. Rukhin, J. Soto, J. Nechvatal, M. Smid, S. Leigh, M. Levenson, M. Vangel, N. Heckert and D. Banks, *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*.

13 F. Pareschi, G. Setti and R. Rovatti, *Implementation and testing of high-speed CMOS true random number generators based on chaotic systems*.

14 S. Petrie and J. Connelly, *A noise-based IC random number generator for applications in cryptography*.

15 Kinniment and E. Chester, Proceedings of the 28th European Solid-State Circuits Conference, pp. 595–598,.

16 M. Bucci, L. Germani, R. Luzzi, A. Trifiletti and M. Varanonuovo, *A high-speed oscillator-based truly random number source for cryptographic applications on a smart card IC*.

17 M. Bucci and R. Luzzi, *Fully digital random bit generators for cryptographic applications*.

18 C. Tokunaga, D. Blaauw and T. Mudge, *True random number generator with a metastability-based quality control*.

19 J. Holleman, S. Bridges, B. Otis and C. Diorio, *A True Random Number Generator With Adaptive Floating-Gate Offset Cancellation*.

20 K. Sathya, J. Premalatha and V. Rajasekar, IOP Conference Series: materials Science and Engineering, pp. 012076,.

21 M. François, D. Defour and C. Negre, *A Fast Chaos-Based Pseudo-Random Bit Generator Using Binary64 Floating-Point Arithmetic*.

22 M. Bakiri, C. Guyeux, J.-F. Couchot, L. Marangio and S. Galatolo, *A hardware and secure pseudorandom generator for constrained devices*.

23 M. Barani, P. Ayubi, M. Valandar and B. Irani, *A new Pseudo random number generator based on generalized Newton complex map with dynamic key*.

24 X. Huang, L. Liu, X. Li, M. Yu and Z. Wu, *A New Pseudorandom Bit Generator Based on Mixing Three-Dimensional Chen Chaotic System with a Chaotic Tactics*.

25 X. Huang, L. Liu, X. Li, M. Yu and Z. Wu, *A new two-dimensional mutual coupled logistic map and its application for pseudorandom number generator*.

26 M. Murillo-Escobar, C. Cruz-Hernández, L. Cardoza-Avendaño and R. Méndez-Ramírez, *A novel pseudorandom number generator based on pseudorandomly enhanced logistic map*.

27 X. Lv, X. Liao and B. Yang, *A novel pseudo-random number generator from coupled map lattice with time-varying delay*.

28 P. Wallden and M. Doosti, *Quantum Cyber Security, Introduction*.

29 S. Ma, J. Liu, Z. Yang, Y. Zhang and J. Hu, *A pseudo-random sequence generation scheme based on RNS and permutation polynomials*.

30 Y. Zhao, C. Gao, J. Liu and S. Dong, *A self-perturbed pseudo-random sequence generator based on hyperchaos*.

31 M. Ozcanhan, M. Unluturk and G. Dalkilic, *An ultra-light PRNG passing strict randomness tests and suitable for low cost tags*.

32 L. Yu, B. He, L. Liu, S. Qian, Y. Huang, S. Cai, Y. Song, Q. Tang and Q. Wan, *Design and FPGA implementation of a pseudorandom number generator based on a four-wing memristive hyperchaotic system and Bernoulli map*.

33 K. Mandal, X. Fan and G. Gong, *Design and implementation of warbler family of lightweight pseudorandom number generators for smart devices*.

34 S. Alhadawi, M. Zolkipli, S. Ismail and D. Lambić, *Designing a pseudorandom bit generator based on LFSRs and a discrete chaotic map*.

35 M. Alawida, A. Samsudin and J. Teh, *Digital cosine chaotic map for cryptographic applications*.

36 S. Chen, B. Li and C. Zhou, *FPGA implementation of SRAM PUFs based cryptographically secure pseudo-random number generator*.

37 Y. Liu and X. Tong, *Hyperchaotic system-based pseudorandom number generator*.

38 D. Lambić and M. Nikolić, *New pseudo-random number generator based on improved discrete-space chaotic map*.

39 M. Riaz, J. Ahmed, R. Shah and A. Hussain, *Novel secure pseudorandom number generator based on duffing map*.

40 A. Akhshani, A. Mobaraki, S.-C. Lim and Z. Hassan, *Pseudo random number generator based on quantum chaotic map*.

41 S. Kalanadhabhatta, D. Kumar, K. Anumandla, S. Reddy and A. Acharyya, *PUF-based secure chaotic random number generator design methodology*.

42 M. Bera, A. Acín, M. Kuś, M. Mitchell and M. Lewenstein, *Randomness in quantum mechanics: philosophy, physics and technology*.

43 Plotnitsky, *A matter of principle: the principles of quantum theory, Dirac's equation, and quantum information*.

44 H.-K. Lo, M. Curty and K. Tamaki, *Secure quantum key distribution*.

45 J. Solem and L. Biedenharn, *Understanding geometrical phases in quantum mechanics: An elementary example*.

46 W. Killmann and W. Schindler, *ser*, BDI, Bonn.

47 J. Arockiasamy, L. Benjamin and R. Vaidyanathan, *Beyond Statistical Analysis in Chaos-Based CSPRNG Design*.

48 M. Naumenko, *Cryptographically secure Pseudorandom Number Generators*, Charles University.

49 J. Katz, *Introduction to Modern Cryptography, Pseudorandomness*.