Fırat Üniversitesi Deneysel ve Hesaplamalı Mühendislik Dergisi

Kriptografik Rasgele Sayı Üreteçlerinin Güvenlik ve Performans Temelli Tasarım ve Değerlendirme Kriterleri

Ali Murat GARİPCAN[1]*, Ebubekir ERDEM[2]

[1]Bilgisayar Mühendisliği Bölümü, Mühendislik Fakültesi, Munzur Üniversitesi, Tunceli, Türkiye.
[2]Bilgisayar Mühendisliği Bölümü, Mühendislik Fakültesi, Fırat Üniversitesi, Elazığ, Türkiye.
[1]alimuratgaripcan@munzur.edu.tr, [2]aberdem@firat.edu.tr

Geliş Tarihi: 28.03.2025
Kabul Tarihi: 02.07.2025
Düzeltme Tarihi:21.05.2025
doi: https://doi.org/10.62520/fujece.1667526
Araştırma Makalesi

Alıntı: A. M. Garipcan ve E. Erdem,’’Kriptografik rasgele sayı üreteçlerinin güvenlik ve performans temelli tasarım ve değerlendirme kriterleri”, Fırat Üni. Deny. ve Hes. Müh. Derg., vol. 4, no 3, pp. 505-523, Ekim 2025.

Öz

Bu çalışma, kriptografik uygulamalarda kullanılan Rastgele Sayı Üreteçleri (RSÜ) için güvenlik ve performans odaklı seçim ve değerlendirme kriterlerini kapsamlı bir biçimde incelemektedir. Çalışmada, Sözde Rastgele Sayı Üreteçleri (SRSÜ) ile Gerçek Rastgele Sayı Üreteçleri (GRSÜ) arasındaki temel farklılıklar detaylı olarak analiz edilmiştir. Bu analizler kapsamında entropi kaynakları, istatistiksel karakteristikler ve performans metrikleri ile bunların güvenlik üzerindeki etkileri değerlendirilmiştir. GRSÜ’lerin, donanıma bağımlı yapıları sayesinde yüksek güvenlik gereksinimlerini karşılama kapasiteleri; buna karşılık, yazılım tabanlı ve düşük maliyetli çözümler sunan SRSÜ’ler ile karşılaştırmalı olarak ele alınmıştır. Çalışma içerisinde, minimum entropi testlerinden alan ve enerji gereksinimlerine kadar uzanan çeşitli kriterler değerlendirilmiş; kriptografik sistemlerde RSÜ tasarımı ve uygulaması sırasında karşılaşılan temel zorluklara dikkat çekilmiştir. Ayrıca, RSÜ tasarımında karşılaşılan temel güvenlik gereksinimleri ve bu gereksinimlerin pratikte nasıl karşılanabileceği ayrıntılı biçimde tartışılmıştır. Bu kapsamda, SRSÜ’ler, fiziksel süreçlere dayalı GRSÜ’ler ve hibrit RSÜ tasarımları analiz edilerek; bu farklı tasarımların avantajları ve dezavantajları karşılaştırmalı olarak sunulmuştur. Sonuç olarak, bu çalışma RSÜ’ler için güvenlik, performans ve tasarım unsurlarını bir araya getiren bir çerçeve sunarak gelecekteki kriptografik araştırmalar için bir ölçüt oluşturuyor. Böylece, güvenli, verimli ve ölçeklenebilir yeni nesil kriptografik sistemler için bir temel sağlıyor.

Anahtar kelimeler: Bilgi güvenliği, Kriptografi, Rasgele sayı üreteçleri, Rasgelelik, Entropi

*Yazışılan yazar

Firat University Journal of Experimental and Computational Engineering

Security and Performance-Based Design and Evaluation Criteria for Cryptographic Random Number Generators

Ali Murat GARİPCAN[1]*, Ebubekir ERDEM[2]

[1]Department of Computer Engineering, Engineering Faculty, Munzur University, Tunceli, Türkiye.
[2]Department of Computer Engineering, Engineering Faculty, Fırat University, Elazığ, Türkiye.
[1]alimuratgaripcan@munzur.edu.tr, [2]aberdem@firat.edu.tr

Received: 28.03.2025
Accepted:02.07.2025
Revision:21.05.2025
doi: https://doi.org/10.62520/fujece.1667526
Research Article

Citation: A. M. Garipcan and E. Erdem, "Security and performance-based design and evaluation criteria for cryptographic random number generators", Firat Univ. Jour. of Exper. and Comp. Eng., vol. 4, no 3, pp. 505-523, October 2025.

Abstract

This study comprehensively examines security and performance-oriented selection and evaluation criteria for Random Number Generators (RNGs) used in cryptographic applications. The study analyzes the fundamental differences between Pseudo Random Number Generators (PRNGs) and True Random Number Generators (TRNGs). Within the scope of these analyses, entropy sources, statistical characteristics, and performance metrics, as well as their effects on security, are evaluated. The ability of TRNGs to meet high security requirements thanks to their hardware-dependent structures is compared with PRNGs, which offer software-based and low-cost solutions. The study evaluates various criteria ranging from minimum entropy tests to space and energy requirements. It also highlights the fundamental challenges encountered in the design and implementation of RNGs in cryptographic systems. Additionally, the fundamental security requirements encountered in RNG design and how these requirements can be met in practice are discussed in detail. In this context, PRNGs, TRNGs based on physical processes, and hybrid RNG designs are analyzed. The advantages and disadvantages of these different design paradigms are then presented in a comparative manner. In conclusion, this study delivers a framework that unifies security, performance, and design considerations for RNGs, setting a benchmark for future cryptographic research. As a result, this framework forms the foundation for the next generation of secure, efficient, and scalable cryptographic systems.

Keywords: Information security, Cryptography, Random number generators, Randomness, Entropy

*Corresponding author

1. Introduction

In our daily lives, random numbers are frequently used in scientific computation and modeling across various fields, including mathematics, physics, statistics, and even the gaming and entertainment industry. For example, computer games designed with a fixed sequence of scenarios or scenes are often considered unappealing. To ensure that the flow of the game can vary even when the initial conditions remain the same, random numbers are used to introduce different scenarios. Similarly, in games of chance, such numbers are required to distribute cards randomly rather than in a predetermined order, thereby preventing possible cheating. There are also many examples in everyday life where random numbers are applied, including lottery draws, assigning candidate numbers in centralized exams, and determining the movement of pieces in a backgammon game. In some of these applications, a simple coin toss or the outcome of a pair of dice thrown into the air may be sufficient. However, in high-security contexts such as cryptographic applications, simple methods like coin tossing are inadequate in terms of randomness quality. In such cases, much more complex and reliable random number generation methods are required [1–2].

The roots of cryptography, forming the foundation of secure communication system design, can be found not only in areas such as military secrecy, diplomacy, and financial management, but also in various fields where the security of political, industrial, and medical data is critically important. The theoretical and practical foundations of the applications and systems developed for this purpose date back to "*La Cryptographie Militaire"*, published by Auguste Kerckhoffs in 1883. In this study, Kerckhoffs proposed a fundamental assumption: even if all parameters of a cryptographic system are known, except for the key, the system should remain secure. This assumption, which remains valid today, plays a significant role in fulfilling the security requirements of modern cryptosystems. In fact, many of these systems and algorithms adopt publicly known algorithms, following the core principles of Kerckhoffs' axiom.

The ENIGMA machine violated this assumption and remains a striking and instructive example in the history of cryptography. Despite being capable of performing high-level encryption, ENIGMA relied entirely on secrecy for its operation, a strategy that ultimately led to its decryption, and once again highlighted the critical importance of Kerckhoffs' principle. This decryption process, effective enough to change the outcome of World War II, marked the beginning of a new era in modern cryptography. Therefore, the success of encryption systems is not solely determined by their technical complexity, but also by how effectively the keys are protected and managed against potential attacks [3].

Keys, which are typically randomly generated, are fundamental security components that define the operation of cryptographic systems and the transformation processes of data. Regardless of their type, nearly all cryptographic algorithms require the generation and use of at least one secret key consisting of random numbers with robust statistical properties to ensure security. The unpredictability of the random numbers used to secure keys and other critical functions plays a vital role in maintaining system security. Ideally, even if an attacker has complete knowledge of the system, these keys should remain indecipherable. In this regard, a well-designed key generation methodology and the use of high-quality random numbers constitute the cornerstone of a secure cryptographic algorithm [4–5].

In cryptographic applications, random numbers are used in various critical processes, including the generation of public-private key pairs, symmetric keys, one-time pads, stream ciphers, and authentication. Security protocols such as Internet Protocol Security (IPsec), Media Access Control Security (MACsec), and Transport Layer Security / Secure Socket Layer (TLS/SSL) also rely on randomly generated numbers during authentication, key exchange, and secure data transmission. An improperly implemented RNG, even when integrated into a robust cryptographic framework, can compromise the entire security infrastructure by introducing unforeseen vulnerabilities exploitable by attackers. Simply put, random numbers lacking a uniform statistical distribution within a defined range can provide attackers with the necessary leverage to break the system. Therefore, the security of cryptographic systems depends not only on the strength of the algorithms employed but also on the robustness of the key generation mechanism. A cryptographic system is only as resilient to sophisticated attacks as its weakest component [6–7].

In cryptographic systems, key management, including the processes of key generation, distribution, storage, modification, and destruction, plays a central role in maintaining overall security. Therefore, robust security protocols require not only the use of high-quality random numbers but also the implementation of an effective and secure key management strategy. For example, keys should never be extracted from the cryptographic system where they are generated, and they must be created and stored within a secure computing environment protected against unauthorized access. However, securely generating or storing random numbers alone does not guarantee unpredictability. If a key is predictable by an attacker, simply storing it in a secure environment is not sufficient [8–9].

Ideally, regardless of an attacker's level of expertise or computational power, guessing a randomly selected or generated key should be infeasible. When a key is derived from a known dataset, its predictability increases from the attacker's perspective, whereas keys generated using truly random methods are significantly harder to predict. In such cases, the only viable strategy for an attacker becomes a brute-force attack. However, the key's probability space must be large enough that exhaustive search becomes practically infeasible. Nevertheless, if sufficient time and resources are available to test all possible combinations, keys should be periodically refreshed to mitigate brute-force threats. Random keys meeting these essential security requirements are generated through specialized structures known as RNGs, as classified in Figure 1 [10-11].

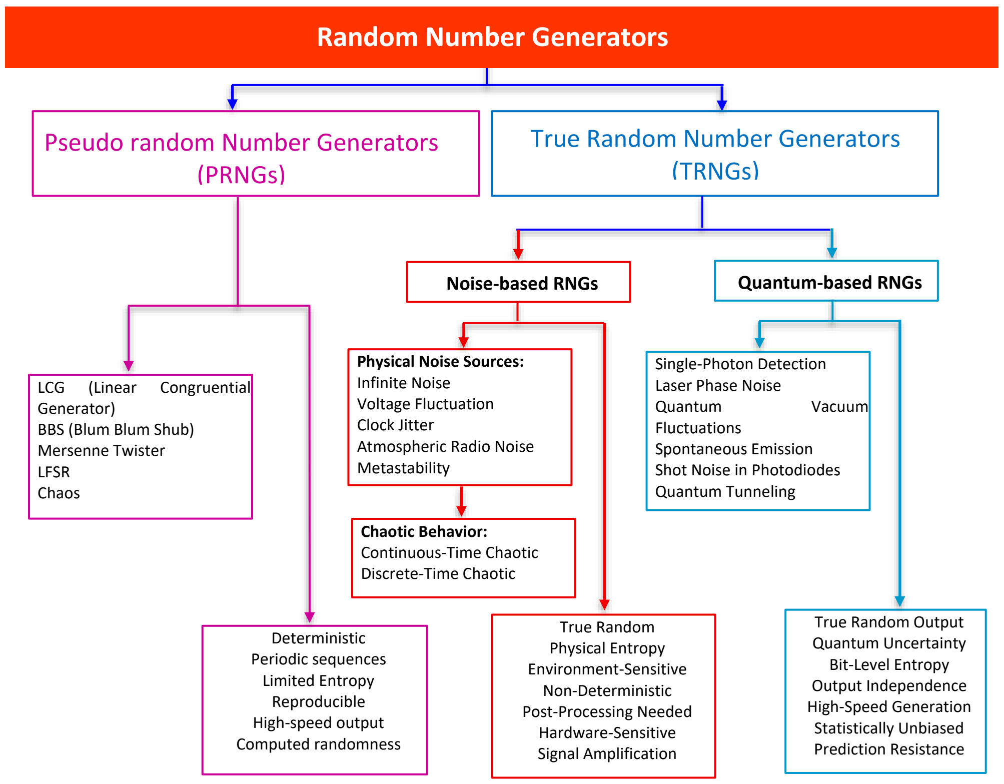

Figure 1. A typical classification of cryptographic RNGs

Due to the high security requirements of cryptographic applications, the RNGs used in such systems are expected to generate random numbers with superior statistical properties that are indistinguishable from those produced by an ideal RNG. If the generated numbers are insufficiently random or demonstrate predictability, they can pose significant risks to the security of cryptosystems. Therefore, a thorough and accurate analysis of the operating principles of RNGs and the security requirements related to randomness is essential, especially for protecting data in open-access digital environments. Otherwise, a predictable relationship between the inputs and outputs of a cryptographic system may enable successful cryptanalysis attacks. For this reason, understanding the fundamental security requirements of cryptographic RNGs and how these requirements can be addressed in practice is critical for the proper evaluation of their design [12–13].

This research paper presents a comprehensive framework for selecting and evaluating cryptographic RNGs, aiming to enhance understanding of the field and provide substantive contributions to the existing scientific literature. Particular attention is given to the distinct design characteristics of PRNGs and TRNGs, along with complex entropy sources and stringent security requirements. Key performance factors influencing RNGs in cryptographic systems are analyzed comprehensively. Serving as a guide to improving security and efficiency in cryptographic infrastructures, the study offers both theoretical and practical insights into developing effective RNG design and evaluation methodologies. From a broader perspective, the proposed framework is expected to contribute to the development of more robust and reliable cryptographic solutions. Analyzing diverse entropy sources and addressing all security challenges associated with various cryptographic RNG types remains beyond the current scope, potentially limiting the generalizability of the findings. Despite the stated limitations, the in-depth contributions focusing on selected RNG types establish a solid foundation for subsequent, broader research in the field.

1.1. Paper organization

The rest of the paper is structured as follows: In the second part, the advantages and disadvantages of cryptographic RNGs and their characteristic differences according to their design types are theoretically discussed. In this context, the security and performance implications of the main differences between PRNGs and TRNGs are analyzed in detail. In the third section, the security and performance criteria required for the selection and evaluation of RNGs for cryptographic purposes are discussed in detail. The last section summarizes the analyses and evaluations carried out in the study and presents the conclusions about the applicability and effectiveness of RNGs in cryptographic systems.

2. Random Number Generators: Design Principles and Structural Differences

Modern cryptographic protocols demand highly reliable RNG solutions due to their increasingly stringent security requirements. Within this context, an ideal RNG is expected to produce random numbers that are mutually independent and uniformly distributed within a defined interval. Although theoretically achievable, implementing such an RNG in practice remains extremely difficult. Any RNG design should aim to approximate ideal randomness as closely as possible. Meeting this expectation requires careful selection and evaluation, as modern protocols demand robust RNG designs built on these standards [2, 6, 14].

The inherent nature of randomness sources and the varying security levels across cryptographic applications necessitate separate evaluation of TRNG and PRNG security and design requirements. Addressing this issue requires detailed analysis of RNG architecture and security assessment grounded in clearly formulated criteria for each design class. In this framework, cryptographic RNGs must fulfill the strict security requirements summarized in Table 1 [14-16].

Table 1. Core security requirements for cryptographic RNGs

Requirement	Description of Requirement
R1	Random numbers must not exhibit any statistical weaknesses.
R2	If an attacker knows any subsequences of the generated random numbers, it must not be possible to compute or predict their predecessor or successor values with high accuracy.
R3	Previous random numbers cannot be computed or predicted with high accuracy, even if the internal state of the RNG is known.
R4	Next random numbers cannot be computed or predicted with high accuracy, even if the internal state of the RNG is known.

2.1. Deterministic (Pseudo) random number generators

In Figure 2, the general design architecture of a typical PRNG can be described by five different variables: S, R, Ψ, φ ve P_S . The state and output spaces of the generator are represented by finite sets S and R, respectively. The variables Ψ and φ are the output and transition functions of the generator. In the system, a random number sequence $r_1, r_2, \ldots \ldots r_{n-1} \in R$ is obtained from the internal state values $S_1, S_2 \ldots S_{n-1} \in S$. The output function $\Psi : S \rightarrow R$ is used to compute the random number r_n, such that $r_n = \Psi(S_n)$. Simultaneously, the internal state S_n is updated using the transition function $S_{n+1} = \varphi(S_n)$. The initial internal state is computed as $S_1 = \varphi(S_0)$, and in the set of equations, P_S, S_0 represents the probabilistic distribution of the seed value [14, 17-18].

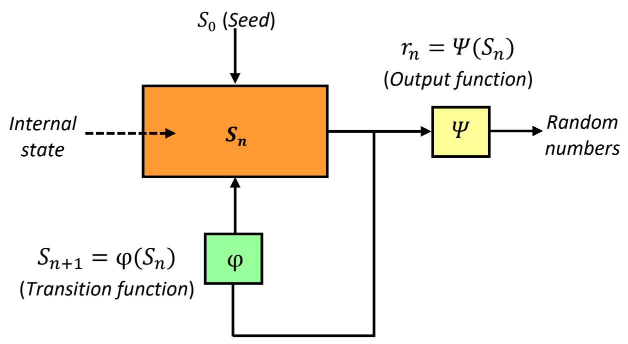

Figure 2. General structure of a PRNG

PRNGs function as deterministic systems grounded in mathematical principles, commonly implemented as finite state machines. In such systems, the internal state or the output at any given time depends on a mathematical function derived from previous states. Therefore, the independence criteria remain only partially satisfied. Being inherently deterministic, PRNGs strictly preserve the entropy level of their initial input without contributing any additional randomness. Hence, the level of randomness (entropy) and unpredictability in the output remains fixed until the seed is modified. In other words, knowledge of the seed value or any internal state in a PRNG enables straightforward prediction of all subsequent states and outputs. This behavior introduces a significant vulnerability when PRNGs are employed in security-critical applications [11].

A well-designed PRNG enables generation of statistically near-perfect, random-looking number sequences at high speeds. However, reusing the same seed value results in identical internal states and, consequently, identical output sequences. Such periodicity represents a major drawback that reduces the applicability of PRNGs in security-critical cryptographic applications. The reliability of a PRNG depends on factors such as algorithmic complexity, period length, and the entropy embedded in the seed value. For example, the maximum period of a PRNG with an internal state length of n bits is limited to 2^n, and selecting an unsuitable seed may significantly shorten this period. Enhancing the cryptographic strength of a PRNG requires increasing the length of its internal state. However, this enhancement incurs additional costs, particularly in hardware-based systems, including greater demand for programmable resources, expanded area requirements, and elevated energy consumption [8, 14, 19].

2.2. True Random number generators

The physical architecture of a TRNG, illustrated in Figure 3, consists of noise source and post-processing unit. The noise sources, forming the non-deterministic part of the system, utilize analog-to-digital conversion and entropy conditioning. Compared to PRNGs, the randomness of the noise sources, along with the digitization and entropy conditioning methods in TRNGs, is highly dependent on the specific hardware employed in the design. For this reason, no universal or standardized definition of TRNGs can be established. In the literature, various physical phenomena such as thermal noise, phase noise (jitter), metastable states, radioactive decay, quantum-based photon interactions, and chaotic circuits serve as commonly used as entropy sources in TRNGs. However, the statistical properties of raw random numbers from the noise source are often insufficient. These flaws can make the numbers predictable to attackers. To ensure cryptographic suitability, TRNGs use an algorithmic post-processing step called entropy conditioning. This step improves the statistical quality of the output.

Compared to PRNGs, TRNGs are preferred for high-security applications. They generate random numbers more slowly and rely on hardware, which makes them computationally expensive. In TRNGs, noise sources are essential for meeting security requirements like unpredictability and non-reproducibility. These sources directly affect the quality of the random numbers generated. When entropy is low, output becomes structured and predictable, a risk known as the statistical weakness problem. To overcome this limitation, post-processing aims to ensure that TRNG outputs meet key cryptographic criteria such as uniform distribution, independence, and high entropy.

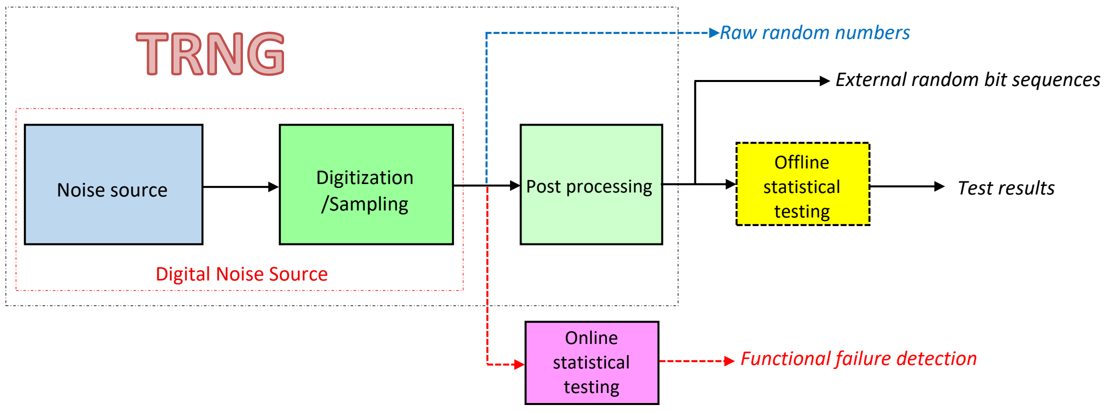
TRNG
Raw random numbers
External random bit sequences
Noise source
Digitization /Sampling
Post processing
Offline statistical testing
Test results
Digital Noise Source
Online statistical testing
Functional failure detection

Figure 3. Typical design blocks of a physical TRNG

2.3. Hybrid random number generators

HRNG architecture combines PRNG reliability with TRNG unpredictability, enhancing state updates with true randomness. HRNGs function as highly reliable PRNG structures leveraging the randomness generated by a TRNG to update both internal states and seed values. The core weakness of standard PRNGs lies in the fact that the system's entire randomness quality is concentrated in a single, potentially predictable seed value. In HRNGs, this tendency toward estimation is significantly reduced through periodic seed updates performed by a TRNG. Moreover, the additional inputs provided by the TRNG, denoted as E_n, not only refresh the seed and internal state values but also contribute to the transition and output functions, φ_H and Ψ_H [16, 20].

Unlike PRNGs, the HRNG architecture illustrated in Figure 4 consists of seven components: $(S, R, E, \varphi_H, \Psi_H, p_S, (q_n)_{n\,\in N})$. This hybrid model integrates deterministic and true randomness components to generate more reliable and unpredictable random numbers. The set E represents the additional input values, while $(q_n)_{n\,\in N})$ denotes the probabilistic distribution of the additional inputs added at step n of the system. For the security of the hybrid architecture, it is important that q_n values are variable. If the additional inputs $(e_1, e_2 \ldots e_n)$ provided to the system remain constant, q_n becomes fixed and no additional randomness contribution can be provided. Under this condition, entropy in the hybrid model depends solely on the seed value, causing the system to behave effectively like a PRNG [21,22].

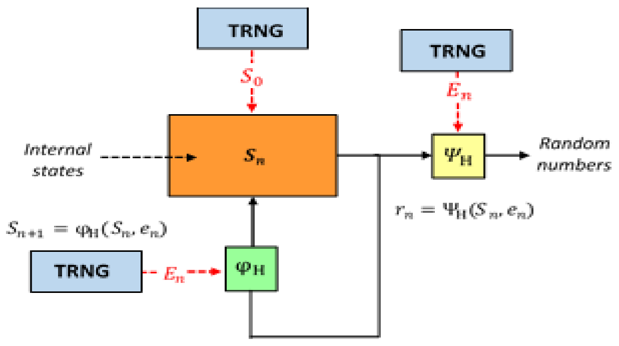

Figure 4. TRNG-based hybrid RNG architecture

2.4. Comparison of PRNGs and TRNGs

The fundamental characteristic requirements for RNGs intended for cryptographic use are listed in Table 1. To operate effectively in sensitive cryptographic systems, RNGs must satisfy four core security requirements: R1, R2, R3, and R4, as defined in Table 1. PRNGs, thanks to their typically strong statistical properties, generally meet requirement R1 successfully. However, this requirement alone is not sufficient, since even partial knowledge of a random number sequence may allow an attacker to infer the entire sequence. Considering the possibility of privileged attackers, the assumption that an adversary knows a specific subset of the number sequence is a highly realistic scenario. For instance, a hybrid system in which confidential data is encrypted using a random session key (k_{rnd}), and this key is delivered to the legitimate recipient via a secure key distribution protocol. Although knowledge of the session key alone may not be sufficient to decrypt all messages, it should be noted that the legitimate recipient in the system may also act as a privileged attacker [23–24].

In PRNGs, all internal states and outputs derive from the seed value, meaning the system's entropy depends entirely on it. This dependency represents one of the primary limitations of PRNGs. For example, in the generator illustrated in Figure 5, which operates based on a recursive relation defined over GF (2), the output sequence is described by the following:

$$c_1, \dots, c_n \in \{0, 1\} \text{ for } a_{n+t+1} \equiv c_1 a_{n+t} \dots c_n a_{n+1} (mod\ 2) \quad (1)$$

In this case, once the initial values (seed values) are known, all subsequent values of the sequence become deterministically computable, and true randomness cannot be achieved.

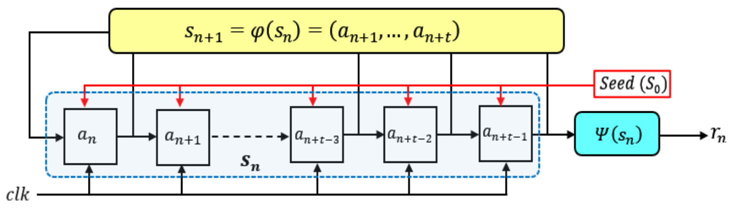

Figure 5. t bit shift register-based PRNG architecture

Based on the formal definition of the internal state values $S = \{0, 1\}^t$, and with the current state defined as $s_n := (a_n, \dots, a_{n+t-1})$, the future states ($s_{n+1}$) and the corresponding output random numbers (r_n) in this LFSR are given by the following expressions:

$$s_{n+1} = \varphi(s_n) = (a_{n+1}, \dots, a_{n+t}) \quad (2)$$

$$r_n = \Psi(s_n) = a_n, r_n \in \{0, 1\} \quad (3)$$

As long as t remains sufficiently large, LFSRs with primitive feedback functions can satisfy the R1 requirement. However, their deterministic nature means that knowledge of an internal state or even part of the output can expose all future values, creating a serious vulnerability. For this reason, LFSR-based generators are unsuitable for use in sensitive cryptographic applications. According to the R2 requirement, the initial seed value S_0 must contain high-entropy random values periodically refreshed by a TRNG and the output (φ) and transition (Ψ) functions of the deterministic RNG should exhibit sufficient complexity. In this context, satisfying both the R1 and R2 requirements is critical to qualifying these generators to be considered suitable for use in security-critical cryptographic applications [14, 16, 23].

In a block cipher system using an encryption function of the form $Enc: \{0,1\}^n \, x \, \{0,1\}^m \rightarrow \{0,1\}^n$, the plaintext and ciphertext spaces are represented by $\{0,1\}^n$ and $\{0,1\}^m$, respectively. As shown in Figure 6, employing block ciphers as RNGs not only satisfies the R2 requirement but can also deliver high-quality random numbers suitable for cryptographic applications. In this context, defining k_{rnd} be the secret key, internal states and random numbers are formally given by $S = \{0,1\}^n \, x \, \{0,1\}^m$ and$R = \{0,1\}^n$, respectively. Based on these formal definitions, the internal state update and random number generation of a block cipher–based RNG are defined as follows:

$$\varphi((Enc(S_n, k_{rnd}), k_{rnd}) = S_{n+1}, \qquad n \geq 0 \tag{4}$$
$$\Psi(Enc(S_n, k_{rnd})) = r_n \tag{5}$$

In the literature, block ciphers exhibiting strong statistical properties are known to satisfy the R1 effectively. However, due to the impracticality of known-plaintext attacks against block ciphers, employing them as RNGs necessitates additional security considerations, particularly regarding unpredictability. Choosing a robust encryption function is crucial to block any attempt to forecast the sequence of generated random numbers. Using well-established block ciphers such as AES and 3DES, together with a securely defined key k_{rnd}, enables block cipher–based RNGs to satisfy not only the R1 requirement for statistical robustness but also the R2 requirement for unpredictability.

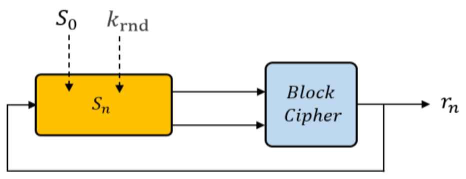

Figure 6. Typical PRNG architecture based on a block cipher

Within the scope of the R3 requirement, any RNG must prevent prediction of its internal states, previously generated random numbers, or past internal values to be predicted. The main purpose of this requirement is to ensure that previously generated values remain incomputable from the current internal state. Accordingly, the R3 requirement necessitates a one-way transition function of the form $\theta: S \rightarrow S$ between the current states of the generator. However, if the value k_{rnd} is known to the attacker,

$$r_{n-1} = Enc^{-1}(r_n, k_{rnd}) \tag{6}$$

Due to the reversible relationship, the R3 requirement cannot be satisfied by the RNG given in Fig. 6. One-way hash functions can be used to fulfill this requirement. For example, in a system where φ and Ψ are identified by 160-bit SHA-1 or RIPEMD hash functions, an RNG identified as $S = R = \{0,1\}^{160}$ can meet the R3 requirement, along with satisfying R1 and R2.

In PRNG s, the complexity of the φ and Ψ functions used for the R3 requirement, along with their mutual interaction, plays a critical role in maintaining security. For example, in a PRNG where both functions are based on the SHA-256 hash function ($\Psi = \varphi = SHA - 256$), and $S = R = \{0,1\}^{256}$, the following structure emerges:

$$S_{n+1} = \varphi(S_n) = \Psi(S_n) = r_n \tag{7}$$

However, R2 and R3 requirements cannot be simultaneously satisfied due to this equality. This is because a one-way relationship between the generated random numbers (r_n) and the future (S_{n+1}) and past (S_n) states of the generator cannot be established based on Eq. 7. This makes it much easier for an attacker to predict random numbers or predict the next random value. Knowledge of any r_n or S_n makes it possible for an attacker to predict the entire sequence of random numbers, such as $r_{n+1}, r_{n+2}, \ldots r_n$. This equality compromises the independence between internal states and output values, enabling attackers to predict subsequent random numbers. As a result, the generator fails to meet forward and backward security, thus violating core requirements of unpredictability in PRNGs.

The R4 requirement in PRNGs can be met by feeding additional input from a cryptographically robust TRNG, increasing true randomness and complexity in the output and transition functions. However, the hardware dependency, higher implementation cost, and relatively low speed of TRNGs introduce integration challenges in practical PRNG systems. Therefore, the unpredictability of these additional inputs becomes essential for satisfying the R4 requirement.

For TRNGs, the R1 and R2 requirements are fundamental design criteria. Unlike PRNGs, TRNGs do not rely on additional mechanisms to satisfy R3 and R4. When the noise source exhibits non-deterministic behavior and the raw random numbers possess sufficient statistical strength, TRNGs are generally accepted to fulfill the R2 requirement. In such scenarios, the R3 and R4 requirements are typically considered inherently satisfied. Based on these considerations, a detailed summary of the comparative characteristics between PRNGs and TRNGs is provided in Table 2.

Table 2. A detailed comparison of PRNGs and TRNGs

Feature	**TRNG**	**PRNG**
Areas of use	Military, financial transactions, and high-security cryptographic applications	Low-security scenarios such as games, simulations, and some cryptographic applications
Source	Entropy from physical processes is used.	Algorithmic or mathematical methods are used.
Implementation	Hardware mandatory	Software-based (hardware optional)
Periodicity	Non-periodic	Periodic
Cost	High due to specialized hardware and physical devices	Very high (algorithmic execution)
Speed	High, depending on the sampling rate	It depends on the speed of physical processes, usually slower.
Predictability	Low (good for encryption)	High, compared to TRNG
Ease of design cycle	Complex	Easy due to their standardized structure
Throughput	Low	High
Theoretical calculation	Constant (time independent)	May change over time
Safety requirement	R1, R2 (defined in standards)	R1, R2, R3, R4 (defined in standards)
Post-processing requirement	Mandatory	Not mandatory
Lifetime	May degrade over time due to hardware aging	Long, depending on software reliability
Cross-platform portability	Difficult for device-specific models	Easy
Unpredictability	Measured by physical phenomena and output entropy	Depends on algorithm strength and seed secrecy

3. Design and Evaluation Criteria for Cryptographic RNGs

The design and evaluation of cryptographic RNGs pose significant challenges, particularly in demonstrating trustworthiness and providing strong supporting evidence. This section presents RNG evaluation principles under two main categories: security metrics and design quality factors. For security evaluation, the focus is on statistical randomness tests and robustness analysis grounded in stochastic modeling and minimum entropy assumptions. Regarding design quality, key criteria include area and energy efficiency, technological compatibility, and output bit rate.

3.1. Security-related evaluation criteria

The use of a cryptographic security protocol alone does not mean that the algorithm is unconditionally secure. The statistical properties of random numbers critically influence the difficulty and impact of potential attacks targeting these protocols. Therefore, instead of attacking a weak cryptographic algorithm directly, adversaries may exploit vulnerabilities in PRNGs or TRNGs. As a result, security remains a central evaluation criterion for cryptographic RNGs and must not be underestimated. In this context, relying solely on statistical analysis of random numbers is insufficient for a comprehensive security assessment. Specifically, aspects such as stochastic modeling, testability, and minimum entropy should be evaluated within a broader framework, supported by valid and verifiable evidence.

3.2. Entropy assumption and security

The trustworthiness of PRNGs depends on the number of trials an attacker would need to guess the seed or any internal state value. It also relies on the complexity of the output and transition functions that hinder such attempts. The use of well-established components such as block ciphers and hash functions, widely regarded as secure, makes these generators practically reliable. However, the security evaluation of PRNGs may change over time, depending on the security level of the cryptographic components used and the success of attacks targeting them. When secure components are used and no known attacks exist, PRNGs are considered secure. In contrast, weak components reduce overall reliability [16, 25–26].

In contrast to PRNGs, the fundamental security assumption in TRNGs relies on the unpredictability of the generated random numbers. This relates directly to the concepts of statistical randomness and entropy. Commonly, the reliability of a TRNG depends on the entropy of the random numbers generated (after appropriate post-processing). Therefore, assuming that entropy estimation is accurate, the theoretical security bound of TRNGs remains time-invariant. Although predicting the output may be theoretically possible, the success rate of such prediction does not improve over time. This is because each output in a TRNG is derived from an independent physical entropy source. Therefore, TRNGs offer a more reliable and time-resilient solution than PRNGs in long-term applications, as they are not exposed to time-dependent predictability risks.

In information theory, the unpredictability of random variables is measured by entropy. From a computational perspective, entropy reflects the expected number of guesses an attacker must make to identify a value, corresponding to the size of the probability space. Theoretical unpredictability is achieved when the entropy of an n-bit random sequence equals n. Among several definitions of entropy, the Rényi entropy [27] given in Eq. 8, where $p_i = Pr(X = x_i)$, provides a general framework that encompasses both Shannon and min-entropy [28].

$$H_a(X) = \frac{1}{1-a}\, log_2\left(\sum_{i=1}^{n} (p_i)^a\right), 0 \leq a \leq \infty \tag{8}$$

Rényi entropy is difficult to calculate in practice. For this reason, two specific cases of Rényi entropy are widely used: minimum entropy and Shannon entropy. Minimum entropy, defined in Eq. 9, represents the limit of Rényi entropy as $\alpha \to \infty$. In such cases, the measurement yields the minimum entropy value.

$$H_{\min}(X) = \lim_{a\to\infty} H_a(X) = -log_2(\max(p_i)) \tag{9}$$

Minimum entropy measures the worst-case uncertainty in a numeric sequence for security applications. It is defined as the negative logarithm of the probability of the most likely output event, as specified in Eq. 7. This entropy level represents the minimum number of attempts an attacker must make to correctly guess the output. A higher value indicates increased unpredictability. The formulation in Eq. 7 assumes independent generation of random variables. This assumption must be carefully evaluated, especially in physical TRNGs, where independence may not be achievable in practice [28–30].

Eq. 10 provides the mathematical definition of Shannon entropy, the most commonly used entropy measure in information theory**.** This definition corresponds to the special case of Rényi entropy with $a = 1$. As in the case of minimum entropy, validating the independence of random variables remains essential when applying Shannon entropy.

$$H_a(X) = -\sum_{i=1}^{n} p_i \, log_2(p_i) \tag{10}$$

In cryptographic applications, it is essential that the output distribution of an RNG approximates an ideal uniform distribution as closely as possible. The probability of observing a specific n-bit random vector is given by:

$$\mathrm{P}_r(X_1 = x_1, X_2 = x_2, \dots., X_n = x_n,) = \frac{1}{2} \tag{11}$$

This implies that all bit combinations are equally likely, under the assumption of uniform and independent distribution. An n-bit vector generated in this manner has a probability of $1/2^n$. For cryptographic security, the total entropy of the vector should be equal to or as close as possible to its length (n). In such a case, the output entropy of the TRNG approaches its maximum value, i.e., 1. High entropy ensures that each bit is unpredictable beyond the probability of $1/2$. This is because the sum of the probability of all evenly and uniformly distributed bits within a defined interval also equals $1/2^n$.

Entropy is a statistical property of a random variable**,** not a directly measurable quantity. Direct measurement is not possible, and statistical tests used for entropy estimation may fail to fully capture the randomness characteristics of the underlying source. This limitation underscores the relevance of test methods such as AIS 20/31 [28] and NIST SP 800-90B [31], which estimate entropy based on stochastic models of noise sources. These tests not only compute the entropy of the generator but also evaluate the reliability of the entropy source. NIST 800-90B emphasizes minimum entropy estimation**,** whereas AIS 20/31 applies a broader methodology that integrates minimum entropy into a comprehensive evaluation framework.

3.3. Statistical randomness assessment

Randomness plays a critical role in modern information technologies, including secure communication, optimization, and solving complex problems. It improves efficiency in these applications while also helping to reduce security vulnerabilities. In optimization methods, randomness is used to explore a wider solution space and to develop more effective solutions. In cryptographic systems, it is applied to ensure both security and confidentiality. Therefore, any RNG must exhibit statistically robust properties to guarantee unpredictability. Attackers may exploit weak statistical characteristics to predict random numbers, posing a significant security risk in cryptographic applications. For this reason, the statistical adequacy of a generator must be verified through valid and reliable testing techniques before it is used.

The main objective of statistical evaluation is to determine whether the generator and its output sequences resemble true randomness. This hypothesis, called null hypothesis (H_0), states that the RNG produces true randomness. Statistical tests based on the characteristics of an ideal RNG can provide strong evidence supporting the validity of this hypothesis. If the null hypothesis H_0 is rejected, meaning the assumption of randomness is not accepted, then the alternative hypothesis (H_a) is adopted. Each test attempts to validate H_0. It does so by comparing a relevant randomness statistic with the reference values expected from an ideal RNG. Under the randomness assumption, the test statistic calculated from the generated sequences is expected to follow a specific distribution. The theoretical reference distribution of this statistic, under H_0, is determined using mathematical methods. A significance level is assigned to the reference distribution and compared with the calculated test statistic. When the test statistic falls below this level, H_0 is rejected and H_a is accepted [9, 13].

In statistical tests, false acceptance or rejection cases, also called Type I and Type II errors in randomness evaluation, may occur, as shown in Table 3. A Type I error refers to rejecting the null hypothesis (H_0) even though the tested sequence is random. This may lead to the incorrect conclusion that the generator does not produce truly random numbers. Similarly, when the tested sequence is not random and the null hypothesis H_0 is still accepted, this results in a Type II error. Such errors may result in incorrect judgments about the generator, especially when based only on test results [22,32–33].

Table 3. Classification of Type I and Type II errors in statistical tests

True State of the Tested Sequence	$\boldsymbol{H_0}$ accepted	$\boldsymbol{H_0}$ rejected
The tested sequence is random	No error	Type I error
The sequence tested is not random	Type II error	No error

In statistical testing, the parameters a and β represent the probabilities of Type I and Type II errors, respectively. Due to these limitations, statistical randomness tests cannot definitively confirm whether the tested number sequences are truly random. In hypothesis testing, the parameter a, or significance level, represents the probability of making a Type I error. Another important value used in each test criterion is the p_{value} which represents the calculated test statistics. This statistic shows the likelihood that the tested sequence is less random compared to an ideal generator. Acceptance of the null hypothesis H_0 for any given test criterion depends on whether the calculated p_{value} is greater than or equal to the predefined significance level α. If this condition fails, H_0 is rejected and the alternative hypothesis H_a is accepted, suggesting that the tested sequence may not be random [33].

In cryptographic applications, the typical value of the significance level α is chosen from the range $[0.001 - 0.01]$. For example, when $a = 0.001$, if the condition $p_{value} \geq a$ is satisfied for tested sequence, the randomness assumption is accepted with 99.9% confidence. This means an error may occur in only 1 out of every 1,000 sequences generated by the RNG. When $p_{value} < a$, tested sequence is considered not random with the same confidence level.

In the literature, there are several statistical test tools designed to analyze the probabilistic properties of random number sequences. These include FIPS 140-2 [34], Dieharder [35], TestU01 [36], AIS 20/31 [28], Knuth's test suite [37], and NIST SP 800-22 [33]. These tools search for detectable patterns in the tested sequences that may indicate non-randomness. Statistical testing includes an error margin and must be evaluated accordingly.

No finite test set can confirm whether a sequence is truly random. Different test tools use distinct statistical distributions and models, and they may produce conflicting results on the same sequence. In other words, a sequence may pass one method and fail under another. The reverse may also occur. These observations show that, although statistical analysis is essential for RNG evaluation, it does not provide complete assurance. For this reason, test results require careful interpretation, and reliable and consistent methods should be preferred when testing RNGs or their outputs.

Statistical tests compare the empirical distribution of RNG outputs with a theoretical distribution. However, they do not confirm the source of randomness or the generation method. These tests can detect certain patterns or anomalies but cannot reveal all weaknesses. Therefore, while valuable, statistical tests remain insufficient for fully validating an RNG. Using multiple tests together supports a more thorough evaluation. Post-processing methods applied to improve statistical output may hide functional issues in the noise source. Although these methods can increase entropy, they may also obscure critical flaws. Effects of post-processing must be analyzed carefully. For TRNGs, the behavior of the noise source requires advanced analysis methods. Evaluation must include raw outputs to accurately determine the level of inherent randomness [17, 23].

3.4. Stochastic modeling

The ability to generate random numbers with a specific entropy level is a key feature of TRNGs. From an external observer's perspective, entropy means that the next bit produced by the TRNG cannot be predicted with certainty beyond the specified entropy level. This holds true regardless of the design, computational capacity, or cryptanalytic knowledge. In PRNGs, computational security can be evaluated independently of implementation details. However, TRNGs may show varying randomness across hardware platforms, even with the same design methodology. This variation results from the differing characteristics of electronic components due to manufacturing and operational conditions. In the absence of a universal standard, TRNGs may rely on diverse physical phenomena and technological features that significantly affect randomness. Therefore, the standardized evaluation framework is necessary to confirm that TRNGs generate output with the required entropy level [8, 13, 24].

A common approach to entropy estimation in TRNGs uses a predefined set of statistical tests. These tests assume that entropy belongs to the noise source itself, not to the random variables derived from it. Mathematically, random numbers are described as sequences of random variables that reflect the noise source behavior. Accordingly, the entropy of these numbers matches the statistical properties of the corresponding sequences. Such tests look for patterns or regularities that may allow partial prediction of the output [21, 28, 38].

A sequence with a given entropy level cannot be fully described by regular patterns. Therefore, statistically validating entropy also depends on demonstrating the absence of deterministic structure. However, any finite test suite can only evaluate a limited set of pattern types, whereas there are infinitely many ways to approximate or reveal structure in a random variable’s distribution. As a result, statistical tests that ignore the behavior of the noise source may overestimate the true quality of randomness in TRNGs. Currently, there is no statistical test suite capable of reliably estimating entropy without relying on stochastic assumptions about the underlying distributions [14–15, 28].

A well-defined stochastic model guaranteeing a minimum entropy bound for random variables directly derived from the noise source can enhance both the practicality and interpretability of statistical tests. Stochastic modeling describes how a physical TRNG operates and captures the distribution of raw or post-processed random variables. The main goal is to characterize the unpredictability of output bits, using forms like $Pr(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$. Depending on whether the output is uniform and independent, Shannon entropy, minimum entropy, or conditional entropy measures may be used. Beyond characterizing unpredictability, stochastic modeling also facilitates the interpretation of statistical test results [17].

3.5. Design-related evaluation criteria

The selection of a cryptographic algorithm depends on the required security level, which is determined by the sensitivity of the information. It is also influenced by implementation factors such as physical integration and cost. These algorithms can be implemented in both software and hardware layers. Hardware-based implementations may achieve higher performance than software-based versions, considering computational cost, power consumption, and processing speed under specific device architectures.

The core concern of cryptographic algorithm security focuses on protecting secret keys. As a critical component of system security, these algorithms should not run in uncontrolled environments, and secret keys must remain within the system boundaries. In modern cryptographic systems, integrating components such as RNGs into a single chip improves overall security. At this stage, digital logic platforms such as Field Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs) are preferred for crypto-hardware implementations, as they enable secure computation environments. FPGAs offer more flexibility than ASICs due to their reconfigurable architecture, allowing modifications to internal connections, system parameters, switches, and look-up tables. Additional features such as simple modeling, high parallelism, fast operation, and support for physical randomness make FPGAs suitable for TRNG designs [11, 39].

In hardware implementations, the performance of electronic circuits is typically evaluated based on factors such as maximum operating frequency, output speed, and power consumption, which influence latency between system and circuit elements. In cryptographic hardware, these basic criteria are accompanied by security requirements, including resistance to physical tampering and environmental threats. For this reason, all relevant factors require attention during hardware design to maintain robustness and defend against attacks. Additionally, increasing component density and improving energy efficiency can enhance system performance and usability.

3.6. Area-energy requirement

Modern FPGAs are integrated circuits built on semiconductor technology. Users configure them to perform specific logic operations. These chips contain hundreds of thousands of configurable logic blocks arranged in a matrix structure. Logic functions within each block use look-up tables (LUTs) and flip-flops. The interconnections and behavior of these blocks can be defined using hardware description languages (HDLs). Beyond basic logic operations, FPGAs support integration of digital signal processing (DSP) units, memory components (RAM and ROM), central processing unit (CPU), and various specialized modules such as phase-locked loops (PLLs), delay-locked loops (DLLs), and arithmetic units capable of floating-point operations [12–14].

Although the theoretical foundation of randomness and the security requirements of cryptographic RNGs are well established, implementing these generators on resource-restricted hardware remains challenging [24]. In practice, cryptographic RNGs are expected to maintain strong statistical properties while minimizing cost, energy consumption, and hardware usage. For this reason, implementation strategy, resource efficiency, and energy optimization are key considerations in RNG design.

For RNGs, security is the primary evaluation criterion. Additional parameters such as output bit rate, area and energy efficiency, and low latency also play an important role. In FPGA-based implementations, the required chip area, typically measured by the number of programmable logic blocks, directly affects feasibility. The available logic resources vary between different FPGA models and manufacturers, which affects the practical implementation of RNGs [39–40].

Designs with high hardware complexity increase demand for programmable logic blocks, leading to greater latency and energy consumption. This situation creates difficulties in TRNG and PRNG implementations on resource-restricted platforms. Continuous-time chaos-based RNGs, due to their complex dynamics, illustrate this challenge clearly. On FPGAs, these generators may consume a large portion of available logic and raise power usage. While FPGAs support efficient computation of chaotic algorithms through reconfigurable logic, this flexibility can increase design cost through greater area and energy requirements.

Discrete-time chaotic systems are often preferred in RNG design because of their simpler mathematical models. These designs commonly use optimized IP cores to handle arithmetic and trigonometric operations over floating-point data. Depending on the mathematical form of the system, such cores can introduce hardware and energy overhead**,** which increases the total cost of randomness generation. In addition, different clock frequencies and internal delays among these cores complicate synchronization, resulting in higher latency and area-energy usage. These issues reduce the output rate and are particularly critical in TRNG and PRNG designs that rely on continuous-time chaotic systems governed by differential equations or on complex post-processing units to ensure security.

3.7. Hardware constraints and platform portability

On FPGA platforms, the design process **includes four main stages**: design entry, synthesis, simulation, and implementation. Each stage requires a specific toolchain. In design entry, the circuit is modeled using schematic design or Hardware Description Languages (HDLs). This is done with compilers such as ISE, Vivado, or Quartus. During synthesis, a netlist is generated by mapping the HDL code to digital logic based on the available resources of the target FPGA. The implementation stage consists of two steps: mapping and placement-routing. In mapping, netlist components are assigned to the logic elements of the FPGA. Placement and routing define the physical location and connections of these blocks on the chip. These steps can be executed automatically by the compiler or manually. Manual placement, however, may lead to dependency on a specific FPGA architecture, reducing portability across platforms.

FPGA-based RNG implementations are generally simpler than those using ASIC architecture. ASIC-based designs often include analog-to-digital converters and other analog components that are unavailable or limited to certain FPGA device families [15]. This situation makes FPGA brand and model selection a critical constraint in TRNG design, especially during implementation. Within the same brand, functional blocks such as trigonometric units or embedded memory may exist only in specific models. Components like PLLs, DLLs, and user defined I/O pins are usually limited in number, further restricting design scalability. This variation affects portability across FPGA platforms. To improve practicality and deployment flexibility, RNG designs should follow general principles that support compatibility across multiple devices. For developing device-independent and portable RNGs, using automatic placement and routing tools offered by the compiler is recommended.

3.8. High and stable output bit rate

Speed is a critical metric for modern telecommunication systems that require cryptographic solutions to ensure secure, high-throughput communication. Generators with low output bit rates cause inefficiencies, particularly in applications requiring rapid key exchange within microseconds. For instance, although the One-Time Pad (OTP) offers theoretical absolute security, it is considered impractical in large-scale dynamic networks. This is because it demands very high key generation and distribution speed. As a result, modern cryptography typically employs symmetric (e.g., AES, DES), asymmetric (e.g., RSA, ECC), and computationally secure primitives such as hash functions (e.g., SHA-256, MD5), which offer more practical performance. From a cryptographic standpoint, generator speed is not a trivial matter. It is a key performance criterion, second only to security. Achieving the right balance between performance and security in RNG design ensures both operational efficiency and secure cryptographic functionality.

In the literature, RNG designs achieve output bit rates ranging from several hundred kilobits to multiple gigabits per second. High-performance security systems, including modern telecommunications servers, often require key updates at the microsecond scale. For instance, a 10-Gbit Ethernet server may require around 20 megabits of randomness per second to generate a 128-bit session key for every 64-kilobit data block. This provides resistance to side-channel attacks. Supporting such requirements depends on delivering both high and stable output bit rates [11, 17].

Bit rate stability is a key factor**,** especially in TRNGs that use post-processing to address statistical limitations of entropy-constrained noise sources. In many applications, maintaining security takes priority over speed. Although complex post-processing algorithms enhance randomness**,** they may reduce throughput considerably**.** This situation introduces a trade-off between performance and security.

Conversely, lightweight post-processing techniques may preserve speed but can cause unstable output rates. For instance, the von Neumann corrector [41], a well-known post-processing method, generates output only from specific input patterns: “01” is interpreted as ‘1’ and “10” as ‘0’, while “00” and “11” are discarded. Although this approach successfully removes bias, it may cause significant output delays when the entropy source produces long runs of identical bits, such as sequences consisting entirely of 0s or 1s. In such situations, no output is produced, which reduces the bit rate and deviates from the statistical definition of randomness. Therefore, in TRNG design, post-processing methods should be evaluated not only for their cryptographic robustness but also for their impact on output stability. Selecting post-processing methods that are both lightweight and secure is necessary for achieving an optimal trade-off between efficiency and statistical quality in RNGs.

4. Conclusion and Recommendations

This study presents a structured classification of cryptographic RNGs, including the strengths and limitations of major design categories. For each category, evaluation criteria based on design and security aspects are discussed, along with their applicability in practical systems. The analysis focuses on key issues such as security testing, minimum entropy, and robustness. It also addresses performance-related factors such as output bit rate, implementation feasibility, and area-energy efficiency. Core security requirements are identified**,** and their relevance to cryptographic use cases is examined**.** Critical parameters, including entropy level, unpredictability, stochastic modeling, and minimum entropy, are analyzed. The focus is on how these parameters impact RNG robustness. The discussion also covers statistical testing methods and compliance with security standards, which are essential in RNG evaluation. In addition, implementation challenges and hardware dependencies in FPGA**-** and ASIC-based designs are analyzed**,** with attention to hybrid solutions.

This research offers a comprehensive framework for designing and evaluating cryptographic RNGs, with emphasis on entropy-focused analysis and security testing. The proposed approach supports the development of secure and efficient RNG architectures, contributing to stronger cryptographic system resilience. These findings provide a foundation for future work on RNG optimization and guide researchers and practitioners developing cryptographically secure solutions.

Considering recent developments, future studies may particularly benefit from exploring artificial intelligence and machine learning–based evaluation frameworks. These techniques have shown promising potential in modeling entropy behavior, detecting anomalies, and adaptively testing TRNG/PRNG systems in real-time environments. Incorporating such intelligent mechanisms can significantly improve both the security assurance and operational reliability of next-generation RNG architectures. By offering a comprehensive framework for RNG design and evaluation, this work aims to contribute to the development of more reliable, portable, and high-performance cryptographic RNGs.

5. Author Contribution Statement

Author 1 contributed to writing, reviewing, and final editing, while Author 2 prepared the initial draft of the manuscript.

6. Ethics Committee Approval and Conflict of Interest

“There is no conflict of interest with any person/institution in the prepared article”

7. Ethical Statement Regarding the Use of Artificial Intelligence

During the writing process of this study, the artificial intelligence tool "ChatGPT," developed by "OpenAI," was used only for limited purposes of linguistic editing and translation. The scientific content, analyses, and results belong entirely to the authors.

8. References

[1] A. Saini, A. Tsokanos, and R. Kirner, “Quantum randomness in cryptography: A survey of cryptosystems, RNG-based ciphers, and QRNGs,” Information, vol. 13, no. 8, pp. 350–358, 2022.
[2] A. Karakaya, Y. Canbay, H. İ. Bülbül, T. Tuğlular, E. Irmak, E. Kavun, and H. Takçı, Cyber Security and Defense: Biometric and Cryptographic Applications. Ankara, Turkey: Nobel Academic Publishing, 2020.
[3] M. Tehranipoor, N. N. Anandakumar, and F. Farahmandi, Hardware Security Training, Hands-On!. Cham, Switzerland: Springer, 2023.
[4] L. Yao, X. Wu, and H. Zhang, “DCDRO: A true random number generator based on dynamically configurable dual-output ring oscillator,” Integration, vol. 93, p. 102053, 2023.
[5] A. Bikos, P. E. Nastou, G. Petroudis, and Y. C. Stamatiou, “Random number generators: Principles and applications,” Cryptography, vol. 7, no. 4, p. 54, 2023.
[6] Y. Yu, “Design and security analysis of TRNGs and PUFs,” Ph.D. dissertation, KTH Royal Inst. of Technology, Stockholm, Sweden, 2022.
[7] S. Yakut, “Random number generator based on discrete cosine transform based lossy picture compression,” NATURENGS, vol. 2, no. 2, pp. 76–85, 2021.
[8] O. Petura, “True random number generators for cryptography: Design, securing and evaluation,” Ph.D. dissertation, Univ. de Lyon, Lyon, France, 2019.
[9] K. Lee, S. Y. Lee, C. Seo, and K. Yim, “TRNG (true random number generator) method using visible spectrum for secure communication on 5G network,” IEEE Access, vol. 6, pp. 12 838–12 847, 2018.
[10] Y. Chen, H. Liang, L. Zhang, L. Yao, and Y. Lu, “High throughput dynamic dual entropy source true random number generator based on FPGA,” Microelectron. J., 2024.
[11] A. Saini, A. Tsokanos, and R. Kirner, “Quantum randomness in cryptography: A survey of cryptosystems, RNG-based ciphers, and QRNGs,” Information, vol. 13, no. 8, pp. 358–375, 2020.
[12] A. J. Acosta, T. Addabbo, and E. Tena-Sánchez, “Embedded electronic circuits for cryptography, hardware security and true random number generation: An overview,” Int. J. Circuit Theory Appl., vol. 45, no. 2, pp. 145–169, 2017.
[13] S. Yakut, T. Tuncer, and A. B. Özer, “A new secure and efficient approach for TRNG and its post-processing algorithms,” J. Circuits, Syst. Comput., vol. 29, no. 15, p. 2050244, 2020.
[14] F. Ozkaynak, “Cryptologic random number generators,” Türkiye Bilişim Vakfı J. Comput. Sci. Eng., vol. 8, no. 2, pp. 37–45, 2015.
[15] V. Fischer, and F. Bernard, “True random number generators in FPGAs,” in Security Trends for FPGAs: From Secured to Secure Reconfigurable Systems. New York, NY, USA: Springer, 2011, pp. 1–20.
[16] W. Schindler, “Random number generators for cryptographic applications,” in Cryptographic Engineering. New York, NY, USA: Springer, 2009, pp. 5–23.
[17] V. Fischer, M. Deutschmann, S. Lattacher, and G. Battum, “Report on selected TRNG and PUF principles,” HECTOR Project, Tech. Rep. D2.1, Univ. Jean Monnet, 2016.
[18] J. D. Golic, “New methods for digital generation and postprocessing of random data,” IEEE Trans. Comput., vol. 55, no. 10, pp. 1217–1229, 2009.
[19] S. Yakut, T. Tuncer, and A. B. Ozer, “Secure and efficient hybrid random number generator based on sponge constructions for cryptographic applications,” Elektronika ir Elektrotechnika, vol. 25, no. 4, pp. 40–46, 2019.
[20] Ö. Aydın, and C. Kösemen, “XorshiftUL+: A novel hybrid random number generator for Internet of Things and wireless sensor network applications,” Pamukkale Univ. J. Eng. Sci., vol. 26, no. 5, pp. 953–958, 2020.
[21] R. B. Naik, and U. Singh, “A review on applications of chaotic maps in pseudo-random number generators and encryption,” Ann. Data Sci., vol. 11, no. 1, pp. 25–50, 2024.
[22] S. Ruhault, “Security analysis for pseudo-random number generators,” Ph.D. dissertation, Ecole Norm. Supérieure (ENS), Paris, France, 2015.
[23] J. Balasch, F. Bernard, V. Fischer, M. Grujić, M. Laban, and O. Petura, “Design and testing methodologies for true random number generators towards industry certification,” in Proc. IEEE 23rd Eur. Test Symp. (ETS), 2018, pp. 1–10.

[24] M. Stipčević, and Ç. K. Koç, “True random number generators,” in Open Problems in Mathematics and Computational Science. Cham, Switzerland: Springer, 2024, pp. 275–315.
[25] F. Yu, L. Li, Q. Tang, S. Cai, Y. Song, and Q. Xu, “A survey on true random number generators based on chaos,” Discrete Dyn. Nat. Soc., vol. 2020, no. 5, pp. 1–10, 2020.
[26] M. C. Belhadjoudja, “Chaos synchronization using nonlinear observers with applications to cryptography,” arXiv preprint arXiv:2108.02577, 2022.
[27] A. R´enyi, “On measures of entropy and information,” in Proc. 4th Berkeley Symp. Math. Statist. Prob., 1961, vol. 1, pp. 547–561.
[28] W. Killmann, and W. Schindler, “A proposal for functionality classes for random number generators,” BDI, Bonn, Germany, Rep., 2011.
[29] W. Schindler, and W. Killmann, “Evaluation criteria for true (physical) random number generators used in cryptographic applications,” in Proc. Cryptograph. Hardw. Embedded Syst. (CHES), 4th Int. Workshop, 2002, pp. 431–449.
[30] V. Fischer, “A closer look at security in random number generators design,” in Proc. Constructive Side-Channel Anal. Secure Design (COSADE), 3rd Int. Workshop, Darmstadt, Germany, 2012, pp. 167–182.
[31] M. S. Turan, E. Barker, J. Kelsey, K. A. McKay, M. L. Baish, and M. Boyle, “Recommendation for the entropy sources used for random bit generation,” NIST Special Publ., vol. 800-90B, 2018.
[32] B. Yang, “True random number generators for FPGAs,” Ph.D. dissertation, Dept. Elect. Eng., Arenberg Doctoral School, Leuven, Belgium, 2018.
[33] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, N. Heckert, J. Dray, S. Vo, and L. Bassham, “SP 800-22 rev. 1a: A statistical test suite for random and pseudorandom number generators for cryptographic applications,” Nat. Inst. Standards Technol. (NIST), Gaithersburg, MD, USA, 2011.
[34] W. M. Daley, C. Shavers, and R. Kammer, “Security requirements for cryptographic modules,” BOOZ-Allen and Hamilton Inc., McLean, VA, USA, Tech. Rep., 1999.
[35] R. G. Brown, “Dieharder: A random number test suite,” 2013. [Online]. Available: https://webhome.phy.duke.edu/~rgb/General/dieharder.php. [Accessed: Mar. 16, 2023].
[36] P. L’Ecuyer, and R. Simard, “TestU01: A C library for empirical testing of random number generators,” ACM Trans. Math. Softw., vol. 33, no. 4, pp. 22–63, 2007.
[37] D. E. Knuth, The Art of Computer Programming, vol. 2, 3rd ed. Boston, MA, USA: Addison-Wesley, 1999.
[38] K. Gołofit, P. Z. Wieczorek, and M. Pilarz, “A chaos-metastability TRNG for natively flexible IGZO circuits,” AEU-Int. J. Electron. Commun., vol. 170, p. 154835, 2023.
[39] M. Bakiri, C. Guyeux, J. F. Couchot, and A. K. Oudjida, “Survey on hardware implementation of random number generators on FPGA: Theory and experimental analyses,” Comput. Sci. Rev., vol. 27, pp. 135–153, 2018.
[40] F. Frustaci, F. Spagnolo, S. Perri, and P. Corsonello, “A high-speed FPGA-based true random number generator using metastability with clock managers,” IEEE Trans. Circuits Syst. II, Exp. Briefs, 2022.
[41] J. von Neumann, “Various techniques used in connection with random digits,” in Collected Works, vol. 5. New York, NY, USA: Pergamon, 1963, pp. 768–770.