Journal of Computer Science
https://dergipark.org.tr/en/pub/bbd

ISSN,e-ISSN: 2548-1304
Volume:IDAP-2022, pp:80-85, 2022
https://doi.org/ 10.53070/bbd.1173694
Research Paper

Bir Sözde Rassal Sayı Üreticinin Parçacık Sürü Optimizasyonu Yöntemi Kullanılarak Gerçek Zamanlı Optimizasyonu

Real-Time Optimization of a Pseudo-Random Number Generator Using Particle Swarm Optimization Method

Muhammed Saadetdin KAYA [*1], Kenan İNCE [1]

[1]Bilgisayar Mühendisliği Bölümü, İnönü Üniversitesi, Malatya, Türkiye

(saadetdin.kaya@gmail.com, kenanince@gmail.com)

Received:Sep.08,2022 Accepted:Sep.16,2022 Published:Oct.10,2022

Özetçe— Sistem tasarımı ve kriptografik yöntemler için kritik bir öneme sahip olan rassal sayı üretimi; işlem gücü yüksek bilgisayarların ortaya çıkmasıyla güvenlik açısından daha da ön plana çıkmaktadır. Bu problemin çözülmesi için fiziksel bir işleyiş ile rassal sayı üretimini hedefleyen gerçek rassal sayı üreteçleri kullanılabileceği gibi yazılım tabanlı olduğu için uygulanması daha kolay olan sözde rassal sayı üreteçleri (SRSÜ) de kullanılabilmektedir. SRSÜ, genellikle bilinen bir algoritmaya sahip olmaları ve aynı şartlar altında tahmin edilebilen sonuçlar vermeleri sebebiyle gerçek manada rassallık sağlayamamaktadırlar. Nitekim çeşitli rassallık şartlarını sağlamaları, sayı üretim hızı ve maliyet gibi sebeplerden dolayı sıkça tercih edilmektedirler. Bu çalışmada, uygulama kolaylığı ve uygulama ortamı sebebiyle tercih edilen bir SRSÜ algoritmasının Parçacık Sürüsü Optimizasyonu (PSO) kullanılarak değişken sistem şartlarında asgari kaynak tüketimi ile azami rassallığa ulaştırılması amaçlanmıştır. Rassallık, Tekrarlama Sınaması ve Sıfır Hipotezi kullanılarak ölçülmüş ve PSO kullanılarak bir SRSÜ'nün optimize edilmesi yoluyla özellikle alan karmaşıklığı açısından ciddi kazanımlar elde edilebileceği sonucuna ulaşılmıştır.

Anahtar Kelimeler : *Sözde Rassal Sayı Üreteci (SRSÜ), Parçacık Sürü Optimizasyonu (PSO), Koşu Testi, Kriptografi*

Abstract— Random number generation, which has a critical importance for system design and cryptographic methods; with the emergence of computers with high processing power, security comes to the fore even more. In order to solve this problem, real random number generators that aim to generate random numbers with a physical operation can be used, as well as pseudo-random number generators (PRNG), which are easier to implement because they are software-based. PRNG cannot provide real randomness because they generally have a known algorithm and give predictable results under the same conditions. As a matter of fact, they are frequently preferred because they provide various randomness conditions, number generation speed and cost. In this study, it is aimed to achieve maximum randomness with minimum resource consumption under variable system conditions by using Particle Swarm Optimization (PSO) of a PRNG algorithm, which is preferred due to its ease of implementation and application environment. Randomness was measured using the Runs Test and the Null Hypothesis, and it was concluded that significant gains can be achieved, especially in terms of space complexity, by optimizing a PRNG using PSO.

Keywords : *Pseudo-random Number Generator (PRNG), Particle Swarm Optimization (PSO), Runs Test, Cryptography*

1. Giriş

Herhangi bir üyesinin, kendisinden öncesi veya sonrası hakkında bilgi vermediği, dizinin geçmişi veya geleceği tahmin edilemeyen rastgele olarak oluşturulmuş sayılar; rassal sayı olarak adlandırılmaktadır. Bu sayılar, kriptografi alanı dışında günümüzde yapay zeka, bilgisayar simülasyonları, istatistik ve optimizasyon vb. birçok farklı alanda doğrudan kullanılmaktadır (Demchik 2011; Downey ve Hirschfeldt, 2010; Sathya vd. 2021). Bu uygulama alanlarının çoğunda, rassal görünen ancak istenildiğinde yeniden üretilebilen sayılar kullanılmaktadır. Bir algoritma tarafından üretilen bu tür sayılara sözde rassal sayılar (SRS) denilmektedir. Bu sayıların rassal olarak oluşturulmasında kullanılan algoritmalara ise rassal sayı üreteci (RSÜ) ismi verilmektedir (Moreau, 1997).

Tamamen öngörülemeyen, yeniden üretilmesi mümkün olmayan ve güçlü istatistiksel karakteristiklere sahip gerçek rassal sayı üreteçlerinin (GRSÜ) aksine uygulama kolaylığı sebebiyle SRSÜ çok daha yaygın bir şekilde kullanılmaktadır. Eski bir geçmişe sahip olmasına rağmen (Galton, 1890; Tippett, 1927) rassal sayılar, rassallığa duyulan ihtiyacın her geçen gün daha da artması sebebiyle gün geçtikçe daha fazla önem kazanmaktadır. Bu çalışmada rassallık ve rassal sayı ifadeleri sözde rassal sayılar özelinde ve kriptografi alanında ele alınmaktadır.

Kriptografide rassal sayılar gizli anahtar oluşturma, başlangıç vektörünün oluşturulması, maskeleme vb. işlemler için yaygın olarak kullanmaktadır. Şifreleme sistemlerini saldırılara karşı daha dayanıklı hale getirmek için kullanılan bu sayıların üretimi için GRSÜ veya SRSÜ kullanılmaktadır. Ancak özellikle Nesnelerin İnterneti konseptinin yükselen trendi (Said ve Masud, 2013) ve bu cihazların kısıtlı donanımsal özellikleri sebebiyle; SRSÜ özellikle hafif sıklet şifreleme algoritmaları için ön plana çıkmaktadır (Poschmann, 2009).

Kriptografik yöntemlerin güvenliği artık büyük ölçüde temelinde yatan algoritmada kullanılan rassal elementlerin sağlığına bağlı olduğundan dolayı, öngörülemeyen ve yüksek entropiye sahip sayı havuzlarının oluşturulması kaçınılmaz bir sorumluluktur. Kullanılan RSÜ'nün zayıf yapısı, sistem için doğrudan bir tehdit olabilmektedir (Dorrendorf vd., 2009). Bu nedenle kullanılan RSÜ'nün sağlığını ölçmek ve saldırılara karşı direncini yorumlayabilmek için çeşitli analiz yöntemleri oluşturulmuş ve SRSÜ'lerin bu testlerin birçoğunu geçemediği görülmüştür (Zaman ve Ghosh, 2012; McCullough, 2006; Luengo ve Villalba, 2021).

Kriptografik sistemlerde sıklıkla kullanılan SRSÜ'ler için kabul edilebilir düzeyde rassal sayı üretimi ihtiyacının karşılanması amacıyla güçlendirme çalışmaları da gün geçtikçe derinleşmektedir (Vigna, 2016; Mishra ve Mankar, 2015; Poorghanad vd., 2008; Bouillaguet vd., 2020). Bu sorunu aşmak için tohum değerinin işlenmesinin değiştirilmesi (Silva vd., 2009), deterministik algoritmaların kullanılması (Murillo-Escobar vd., 2017), hibrit algoritmaların oluşturulması (Wortman vd., 2018) gibi çeşitli yöntemler geliştirilmiştir. Bu yöntemlere ek olarak makine öğrenmesi ve çeşitli optimizasyon yöntemlerinin de kullanılmasıyla birlikte SRSÜ üretimine yeni bakış açıları getirilmiştir (Pasqualini ve Parton, 2020; Park vd, 2022; Almardeny vd., 2022).

Bu optimizasyon yöntemlerinden biri olan Parçacık Sürü Optimizasyonu (PSO) (Clerc, 2010), matematiksel modellemenin tam olarak yapılamadığı sistemlerde, değişken şartlar altında iyi sonuçlar vermesi sebebiyle SSRÜ'lerin optimizasyonu ve yorumlanması aşamalarında sıklıkla kullanılmaya başlanmıştır (Ma ve Vandenbosch, 2012; Pluhacek vd, 2014; Ding ve Tan, 2014).

Bu çalışmada MATLAB (Toolbox, 1993) ortamında, yalın haldeki MATLAB RSÜ (Savicky, 2006) PSO kullanılarak değişken sistem şartları altında optimize edilmiştir. Bu optimizasyon (rassal sayıların kritik önem taşımadığı ancak elzem olduğu işlemler hedeflenerek) sıklıkla kullanılan paket programlardaki RSÜ'lerin algoritmasında herhangi bir değişiklik yapılmadan elde edilebilecek kazanımları görme amacıyla yapılmıştır.

Elde edilen sonuçlar ışığında, paket programlardaki SRSÜ'lerin hangi şartlar altında kullanılabileceği, kullanım esnasında performans artışının sağlanıp sağlanamayacağı ve sisteme olan etkileri gibi sorular cevaplanmaya çalışılmıştır.

2. Sözde Rassal Sayı Üreteçleri

SRSÜ, başlangıç tohum değeriyle rastgele sayı dizileri üreten deterministik algoritmalardır. SRSÜ ile oluşturulan diziler, bir önceki algoritma durumuna bağlılıkları nedeniyle doğaları gereği GRSÜ özelliklerini taşıyamazlar. Nitekim bu diziler, basit yapıları nedeniyle bilimin birçok alanında faydalı olarak kabul edilmekte olup hala kullanılmaktadır (Boyar, 1989).

Geleneksel SRSÜ'ler matematiksel lineer denklemlerle oluşturulmuştur. Ancak bu üreteçler, SRSÜ'lerin rassalık özelliklerinden emin olmak amacıyla kullanılan matematiksel analizlerin ortaya koyduğu bazı kriterleri aşamamaktadır. Bu nedenle SRSÜ için çeşitli yaklaşımlar önerilmiştir. Bu yaklaşımlar:

1. Kaotik Harita Tabanlı SRSÜ
2. Polinomsal Eşitlik Tabanlı SRSÜ
3. Donanım Tabanlı SRSÜ
4. Doğadan Esinlenilen SRSÜ
5. Kriptografik Algoritma Tabanlı SRSÜ

olarak 5 ayrı kategoride sınıflandırılabilmektedirler.

Bazı bilim insanları, blok şifreleme algoritmaları ile daha güvenli SRSÜ'lerin oluşturulması ile ilgili çeşitli çalışmalar yapmıştır. Bu SRSÜ'lerin belirli güvenlik standartlarını sağlayan, GRSÜ'ye daha yakın rassal sayı dizileri ürettikleri bilinmektedir (Lee vd., 2016). Blok şifreleme algoritmalarının kullanılmasının yanında, bazı araştırmacılar, SRSÜ'nün temelinin sensörler gibi rastgele kaynaklardan gelen tohumlarla inşa edildiği yapılar

geliştirmişlerdir (Hong ve Liu, 2015; Bakiri vd., 2018). Tohum değerlerinin rastgele ve tahmin edilmesinin oldukça zor olmaları sebebiyle bu sensör tabanlı yöntem rassallığı iyileştirmiştir ancak bütünüyle yazılımsal bir yöntem olmadığı için hız konusunda diğer SRSÜ'lerden geride kalmaktadır. Bu nedenle SRSÜ'leri daha verimli hale getirmek ve yazılımsal olarak daha iyi performans göstermek için, rassal sayı üretmek üzere birçok kaotik sistem tabanlı SRSÜ önerilmiştir (Akhshani vd., 2014; Tutueva vd., 2020).

2.1. Sözde Rassal Sayı Üreteçlerinin Analizi:

Yukarıda bahsedilen tüm bu algoritmalar tarafından üretilen rassal sayıların rassallık özelliklerini doğrulamak için çeşitli analiz yöntemleri geliştirilmiştir. Bu analiz yöntemleri temel olarak İstatistiksel Analiz (McCullough, 2006) ve Güvenlik Analizi (Dodis vd, 2013) olarak iki kategoride toplanılabilmektedir. İdeal bir SRSÜ, rassal sayıların dizi içerisindeki düzgün dağılımına ve üretilen sayıların arasındaki bağımsızlığa sahip olmalıdır.

Üretilen dizilerin rassallığını ölçmek için pek çok istatistiksel araç standartlaştırılmıştır. En sık kullanılan istatistiksel test aracı grupları NIST (Rukhin vd., 2002) ve TestU01 (L'ecuyer ve Simard, 2007) bünyesindeki test araçlarıdır. Bu test araçlarının kullanımı ücretsiz olup SRSÜ'lerin rassallığını istatistiksel olarak ölçmek için kullanılmaktadırlar. Her test, oluşturulan rassal sayı dizisinin testi geçip geçmediğinin belirlenmesini sağlayan bir P-değeri üretir. Bir dizinin herhangi bir testi geçmesi o dizinin tamamen güvenli olduğu anlamına gelmemektedir. Bu nedenle dizi tüm testler tarafından değerlendirilmelidir. Paket programlar dahilindeki SRSÜ'ler genellikle bu testlerin birçoğunda başarısız olmakta ve bu nedenle güçlendirilmeye ihtiyaç duymaktadırlar (Savicky, 2006).

Bu test yöntemlerinden biri olan Tekrarlama Sınaması Testi (TST) ilk olarak 1968 yılında Bradley tarafından ortaya atılmıştır (Bradley, 1968). Günümüzde ise hem yalın halde hem de çeşitli varyasyonlarda bir sayı dizisinin rassallığını ölçmek amacıyla kullanılmaktadır (Agin ve Godbole, 1992; Bujang ve Sapri, 2018).

Bir koşu, bir dizi artan değer veya bir dizi azalan değer olarak tanımlanır. Artan veya azalan değerlerin sayısı, koşunun uzunluğudur. Rastgele bir veri setinde (i+1)'inci değer i'inci değerden daha büyük veya küçük olma olasılığı, koşma testinin temelini oluşturan, binom dağılımını takip eder.

TST, test aracı gruplarındaki diğer testlerin aksine yalnızca dizi elemanlarının dizi içerisinde ne kadar sıklıkla bulunduğu ile ilgilendiğinden dolayı SRSÜ'lerin kullanıldığı problemlerdeki yeterlilik şartlarının sağlanıp sağlanamayacağını ölçmek açısından önem arz etmektedir. Bu çalışmada TST, rassal sayı dizisinin potansiyel entropisinden bağımsız olarak dizide bulunan sayıların ne kadar tekrar ettiğini ve bu tekrar sayısının rassallık şartlarını sağlayıp sağlamadığını görmek amacıyla kullanılmıştır.

3. Parçacık Sürü Optimizasyonu

Kendi kendine organizasyon Sürü Zekası (SZ) sisteminin önemli bir özelliğidir. Başlangıçta düzensiz bir sistemin bileşenleri arasındaki yerel etkileşimlerden sürüsel bir düzenin veya koordinasyonun ortaya çıktığı bir süreçtir. Kendiliğinden olan bu süreç sistem içinde veya dışında herhangi bir ajan tarafından kontrol edilmemektedir (Zhang vd., 2014). Kendiliğinden ve kontrolsüz olarak gerçekleşen bu süreç özellikle matematiksel modellemenin yapılamadığı, girdi parametrelerinin belirlenemediği, gerçek dünyadan bilgisayar dünyasına aktarımın zor olduğu karmaşık yapıdaki sistemlerin optimizasyonu için büyük önem arz etmektedir.

Belirgin bir başlangıç değerine veya ajana ihtiyaç duymayan bu yaklaşımın kullanıldığı PSO, aramayı her iterasyonda güncellenen bir parçacık sürüsü aracılığıyla gerçekleştirir. Optimal çözümü aramak için her parçacık, sürüdeki daha önce en iyi konumuna ve küresel en iyi konuma doğru bir maliyet/geri besleme fonksiyonu aracılığıyla belirlenmiş olan bir arama bölgesinde hareket eder. Bu şekilde sistem optimizasyonu gerçekleştirilmiş olur (Zhang vd., 2014).

Bu çalışmada PSO, halihazırda işleyen bir sistemde; değişken sistem şartları altında gerçekleştirilen sözde rassal sayı üretimi işleminin gerçek zamanlı olarak rassal sayı üretimi işleminin optimize edilmesi amacıyla kullanılmıştır. PSO yöntemi; çok değişkenli ve tam olarak belirlenemeyen modellerde diğer optimizasyon yöntemlerine göre daha etkin sonuç vermesi (Zhang vd., 2015) sebebiyle tercih edilmiştir.

4. Uygulama

Uygulama MATLAB (Toolbox, 1993) ortamında, yalın haldeki MATLAB RSÜ (Savicky, 2006) kullanılarak çeşitli aralıklarda rassal sayı dizileri oluşturularak gerçekleştirilmiştir. PSO'nun maliyet fonksiyonunun tanımlanmasında TST kullanılmış ve çıktı parametrelerinden P-değeri kriter olarak belirlenmiştir.

P-değeri, sıfır hipotezi altında gözlemlenen değer kadar aşırı veya ondan daha aşırı bir test gözlemlenme olasılığını ortaya koyacağı için; daha küçük p değerleri sıfır hipotezinin geçerliliği konusunda şüphe uyandıracaktır. Sıfır hipotezi ise (çıkarımsal istatistikte) beklenenin dışında bir durum olmadığını, gruplar ya da değişkenler arasında bir ilişki bulunmadığını veya ölçülen iki olgunun arasında bir fark olmadığını kabul eden genel bir önermedir. Veriler genellikle %5'inden daha az olasılıkla gözlemlenecek şekilde iseler sıfır hipotezinin

yanlış olduğu (sağlanamadığı) çıkarımına varılmaktadır. Bu P-değerinin azami olması ise üretilen rassal sayı dizisinin diğer dizilere göre daha sağlıklı olduğu konusunda yorum yapmamıza olanak sağlamaktadır.

MATLAB iç kütüphanesinde bulunan runstest() fonksiyonun; h çıktı parametresi sıfır hipotezinin sonucunu (0: reddetmemek, 1: reddetmek), p çıktı parametresi ise bu sonucun sağlığını göstermektedir. Bu fonksiyonun sonucunda h çıktı parametresinin 0 olması durumunda TST rassallığının %5 anlamlılık düzeyinde olduğu reddedilmemektedir.

PSO'nun arama bölgeleri; [1:255] ve [1:65535] olarak belirlenmiş olup her arama bölgesinde 10 iterasyon olacak şekilde randi fonksiyonu kullanılarak her iterasyonda başlangıç noktasından bitiş noktasına kadar r adet rassal sayı üretilmiş ve tüm değerler her iterasyonda TST'ye tabii tutularak P-değerleri yorumlanmıştır.

4. Sonuçlar

Uygulamanın [1:255] arama bölgesinde olan bölümü 10 iterasyon olarak çalıştırılıp:

Tablo 1. [1:255] Arama Bölgesi Sonuçları

	Azami P-Değeri	Azami Rassallık İçin Başlangıç Değeri	Azami Rassallık İçin Bitiş Değeri	Bitiş Değerinin Bit Değeri
İdeal	1.00	1	255	11111111
Uygulama	0.6843	1	53	110101

Tablo 1.'deki sonuçlara ulaşılmıştır. İdeal durumda azami entropinin sağlanması için rassal sayı üretimi [1:255] bölgesi arasında yapılmalı iken azami rassallık [1:53] bölgesinde sağlanmıştır. Yine Tablo 1.'de görüleceği üzere azami rassallığın sağlanması için gerekli bit değerinin karakter sayısı 8 bitten 6 bite düşmüştür. Bu da alan karmaşıklığı açısından %25 kazanç anlamına gelmektedir.

Uygulamanın [1:65535] arama bölgesinde olan bölümü 10 iterasyon olarak çalıştırılıp:

Tablo 2. [1:65535] Arama Bölgesi Sonuçları

	Azami P-Değeri	Azami Rassallık İçin Başlangıç Değeri	Azami Rassallık İçin Bitiş Değeri	Bitiş Değerinin Bit Değeri
İdeal	1.00	1	65535	1111111111111111
Uygulama	0.7687	1	13454	11010010001110

Tablo 2.'deki sonuçlara ulaşılmıştır. İdeal durumda azami entropinin sağlanması için rassal sayı üretimi [1:65535] bölgesi arasında yapılmalı iken azami rassallık [1:13454] bölgesinde sağlanmıştır. Yine Tablo 2.'de görüleceği üzere azami rassallığın sağlanması için gerekli bit değerinin karakter sayısı 16 bitten 14 bite düşmüştür. Bu da alan karmaşıklığı açısından %12,5 kazanç anlamına gelmektedir.

Elde edilen sonuçlar araştırmacıların sıklıkla kullandığı bir paket programın iç kütüphanesindeki kaynaklar kullanılarak yapılan bir uygulama sonucunda elde edilmiştir. Bu sonuçlar göz önüne alındığında SRSÜ'lerin optimizasyonu ile alan karmaşıklığı açısından ciddi bir kazanç elde edilebileceğini ortaya koymaktadır. Alan karmaşıklığının yanı sıra döngü kullanılması durumunda iterasyon sayısının azalacağı ve bu sayede zaman karmaşıklığının da yine O(n) olmasına rağmen n değerinin düşmesi sebebiyle pratikte azalacaktır.

PSO'nun uygulamadaki en büyük etkisi, değişken sistem yüklerine hıza adapte olabilmesi ve bu sayede gerçek zamanlı sistemlerde doğrudan kullanılabilecek olmasıdır. İlerleyen çalışmalarda, sürekli olarak rassal sayı üretimine ihtiyaç duyan sistemlerde sayı üretimi için gerekli olan parametrelerin, bir PSO ön katmanı ile elde edilmesi yoluyla rassal sayı üretimi işleminin daha optimize çalışması sağlanabilecektir.

Teşekkür

Bu çalışma, İnönü Üniversitesi Bilimsel Araştırma Projeleri Bölümü'nün (BAPB) FBG-2020-2143 sayılı projesi ile desteklenmiştir. Yazar, değerli geri bildirimleri için İnönü Üniversitesi BAPB'ye teşekkür eder.

Kaynaklar

Agin, M. A., & Godbole, A. P. (1992). A new exact runs test for randomness. In Computing Science and Statistics (pp. 281-285). Springer, New York, NY.

Akhshani, A., Akhavan, A., Mobaraki, A., Lim, S. C., & Hassan, Z. (2014). Pseudo random number generator based on quantum chaotic map. Communications in Nonlinear Science and Numerical Simulation, 19(1), 101-111.

Almardeny, Y., Benavoli, A., Boujnah, N., & Naredo, E. (2022). A Reinforcement Learning System for Generating Instantaneous Quality Random Sequences. IEEE Transactions on Artificial Intelligence.

Bakiri, M., Guyeux, C., Couchot, J. F., & Oudjida, A. K. (2018). Survey on hardware implementation of random number generators on FPGA: Theory and experimental analyses. Computer Science Review, 27, 135-153.

Bouillaguet, C., Martinez, F., & Sauvage, J. (2020). Practical seed-recovery for the PCG pseudo-random number generator. IACR Transactions on Symmetric Cryptology, 175-196.

Boyar, J. (1989). Inferring sequences produced by a linear congruential generator missing low-order bits. Journal of Cryptology, 1(3), 177-184.

Bradley, J. V. (1968). Distribution-free statistical tests.

Bujang, M. A., & Sapri, F. E. (2018). An application of the runs test to test for randomness of observations obtained from a clinical survey in an ordered population. The Malaysian Journal of Medical Sciences: MJMS, 25(4), 146.

Clerc, M. (2010). Particle swarm optimization (Vol. 93). John Wiley & Sons.

Demchik, V. (2011). Pseudo-random number generators for Monte Carlo simulations on ATI Graphics Processing Units. Computer Physics Communications, 182(3), 692-705.

Ding, K., & Tan, Y. (2014, July). Comparison of random number generators in particle swarm optimization algorithm. In 2014 IEEE Congress on Evolutionary Computation (CEC) (pp. 2664-2671). IEEE.

Dodis, Y., Pointcheval, D., Ruhault, S., Vergniaud, D., & Wichs, D. (2013, November). Security analysis of pseudo-random number generators with input: /dev/random is not robust. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (pp. 647-658).

Dorrendorf, L., Gutterman, Z., & Pinkas, B. (2009). Cryptanalysis of the random number generator of the windows operating system. ACM Transactions on Information and System Security (TISSEC), 13(1), 1-32.

Downey, R. G., & Hirschfeldt, D. R. (2010). Algorithmic randomness and complexity. Springer Science & Business Media.

Galton, F. (1890). Dice for statistical experiments. Nature, 42(1070), 13-14.

Hong, S. L., & Liu, C. (2015). Sensor-based random number generator seeding. IEEE Access, 3, 562-568.

L'ecuyer, P., & Simard, R. (2007). TestU01: AC library for empirical testing of random number generators. ACM Transactions on Mathematical Software (TOMS), 33(4), 1-40.

Lee, W. K., Cheong, H. S., Phan, R. C. W., & Goi, B. M. (2016). Fast implementation of block ciphers and PRNGs in Maxwell GPU architecture. Cluster Computing, 19(1), 335-347.

Luengo, E. A., & Villalba, L. J. G. (2021). Recommendations on statistical randomness test batteries for cryptographic purposes. ACM Computing Surveys (CSUR), 54(4), 1-34.

Ma, Z., & Vandenbosch, G. A. (2012, March). Impact of random number generators on the performance of particle swarm optimization in antenna design. In 2012 6th European conference on antennas and propagation (EUCAP) (pp. 925-929). IEEE.

McCullough, B. D. (2006). A review of TESTU01.

Mishra, M., & Mankar, V. H. (2015). Text encryption algorithms based on pseudo random number generator. International Journal of Computer Applications, 111(2).

Moreau, T. (1997). Pseudo-random generators, a high-level survey-in-progress. Published on the internet: http://www. cabano. com/connotech/RNG. HTM.

Murillo-Escobar, M. A., Cruz-Hernández, C., Cardoza-Avendaño, L., & Méndez-Ramírez, R. (2017). A novel pseudorandom number generator based on pseudorandomly enhanced logistic map. Nonlinear Dynamics, 87(1), 407-425.

Park, S., Kim, K., Kim, K., & Nam, C. (2022). Dynamical Pseudo-Random Number Generator Using Reinforcement Learning. Applied Sciences, 12(7), 3377.

Pasqualini, L., & Parton, M. (2020). Pseudo random number generation: A reinforcement learning approach. Procedia Computer Science, 170, 1122-1127.

Pluhacek, M., Senkerik, R., & Zelinka, I. (2014). Particle swarm optimization algorithm driven by multichaotic number generator. Soft Computing, 18(4), 631-639.

Poorghanad, A., Sadr, A., & Kashanipour, A. (2008, December). Generating high quality pseudo random number using evolutionary methods. In 2008 international conference on computational intelligence and security (Vol. 1, pp. 331-335). IEEE.

Poschmann, A. Y. (2009). Lightweight cryptography: cryptographic engineering for a pervasive world. In Ph. D. Thesis.

Rukhin, A., Soto, J., Nechvatal, J., Barker, E., Leigh, S., Levenson, M., ... & Iii, L. E. B. (2002). A statistical test suite for random and pseudorandom number generators for cryptographic applications," NIST Special Publication 800-22 (revised May 15.

Said, O., & Masud, M. (2013). Towards internet of things: Survey and future vision. International Journal of Computer Networks, 5(1), 1-17.

Sathya, K., Premalatha, J., & Rajasekar, V. (2021, February). Investigation of strength and security of pseudo random number generators. In IOP Conference Series: materials Science and Engineering (Vol. 1055, No. 1, p. 012076). IOP Publishing.

Savicky, P. (2006). A strong nonrandom pattern in Matlab default random number generator. Manuscript available from http://www2. cs. cas. cz/~ savicky/papers.

Savicky, P. (2006). A strong nonrandom pattern in Matlab default random number generator. Manuscript available from http://www2. cs. cas. cz/~ savicky/papers.

Silva, R. M., Crespo, R. G., & Nunes, M. S. (2009). LoBa128, a Lorenz-based PRNG for wireless sensor networks. International Journal of Communication Networks and Distributed Systems, 3(4), 301-318.

Tippett, L. H. C. (1927). Random sampling numbers.

Toolbox, S. M. (1993). Matlab. Mathworks Inc.

Tutueva, A. V., Nepomuceno, E. G., Karimov, A. I., Andreev, V. S., & Butusov, D. N. (2020). Adaptive chaotic maps and their application to pseudo-random numbers generation. Chaos, Solitons & Fractals, 133, 109615.

Vigna, S. (2016). An experimental exploration of Marsaglia's xorshift generators, scrambled. ACM Transactions on Mathematical Software (TOMS), 42(4), 1-23.

Wortman, P., Yan, W., Chandy, J., & Tehranipoor, F. (2018). P2M-based security model: security enhancement using combined PUF and PRNG models for authenticating consumer electronic devices. IET Computers & Digital Techniques, 12(6), 289-296.

Zaman, J. K. M. S., & Ghosh, R. (2012). Review on fifteen Statistical Tests proposed by NIST. Journal of Theoretical Physics and Cryptography, 1, 18-31.

Zhang, Y., Agarwal, P., Bhatnagar, V., Balochian, S., & Zhang, X. (2014). Swarm intelligence and its applications 2014. The Scientific World Journal, 2014.

Zhang, Y., Balochian, S., Agarwal, P., Bhatnagar, V., & Housheya, O. J. (2014). Artificial intelligence and its applications. Mathematical problems in Engineering, 2014.

Zhang, Y., Wang, S., & Ji, G. (2015). A comprehensive survey on particle swarm optimization algorithm and its applications. Mathematical problems in engineering, 2015.