World Journal of Advanced Engineering Technology and Sciences

eISSN: 2582-8266

Cross Ref DOI: 10.30574/wjaets

Journal homepage: https://wjaets.com/

(Research Article)

Effectiveness of pseudorandom number generators in secret image retrieval fidelity for steganography

Rawan Khanfar, Hala Ghannam, Alya Alabdouli and Tamer Rabie *

Department of Computer Engineering, College of Computing and Informatics, University of Sharjah, U.A.E.

World Journal of Advanced Engineering Technology and Sciences, 2024, 12(01), 289–297

Publication history: Received on 04 May 2024; revised on 09 June 2024; accepted on 12 June 2024

Article DOI: https://doi.org/10.30574/wjaets.2024.12.1.0238

Abstract

Steganography, an ancient technique for concealing information within seemingly benign data, has resurfaced in the digital age, finding widespread application across a variety of industries. However, its use confronts major hurdles, notably with image files that are susceptible to network transmission distortions and assaults. Pseudorandom Number Generators (PRNGs) emerge as critical safeguards in this setting, increasing the unpredictability of embedded data and thwarting malevolent manipulations. This study evaluates the usefulness of various PRNGs in reinforcing LSB steganography to secure the integrity and retrievability of concealed data after transmission. Six PRNGs, including LCG, PCG, and XORshift, were evaluated, and the results show that they are successful at preventing both distortions and attacks. Notably, non-randomized images of type PNG are resistant to transmission distortions but struggle with secret image retrieval post-attack. This research advances steganographic methodologies, offering insights into fortifying digital communication amidst real-world challenges.

Keywords: Pseudorandom Number Generator; PRNG; Secret Image; Retrieval Fidelity; Steganography; Information Security

1. Introduction

Steganography is the art of hiding information among other non-secret data, as it is as old as writing originally, yet it has come back into vogue in the digital era [1]. It is widely applied in several industries, from multimedia systems to secure communications. Steganography ensures that secret information can be conveyed in plain sight without drawing any attention by embedding concealed messages within photos, videos, or audio files.

There are many challenges and difficulties that threaten steganography, specifically when talking about image files. Generally, images during transmission in the network, are vulnerable, as they are susceptible to compression artifacts, different types of noise, and malicious attacks that cause distortion to the hidden message [2]. When the stego-image is distorted or altered in any way, it becomes very challenging to retrieve the embedded secret. This vulnerability presents a serious risk, particularly in situations when it is imperative to preserve the integrity of hidden data.

To mitigate the stego-images against these attacks and distortions, Pseudorandom Number Generators (PRNGs) have an essential role. The way secret data is embedded into an image can be made more unpredictable by using PRNGs, which are algorithms that use mathematical formulas to produce sequences of random numbers [3]. The main idea is to randomize the confidential data before steganographically embedding it. This will increase the data embedding pattern's resistance to attacks and distortions and make it less predictable. In spite of the typical difficulties experienced during picture transmission, this study attempts to assess how well different PRNGs improve the security of LSB (Least Significant Bit) steganography by guaranteeing that the secret data, once randomized and embedded, stays intact and retrievable post transmission through the network.

* Corresponding author: Tamer Rabie; Email: trabie@sharjah.ac.ae

By exploring different types of PRNGs and studying the interplay between them and the LSB steganography, the goal of this research is to further the development of more resilient digital steganographic methods that can handle the challenges of handling data and communicating in the real world.

2. Mathematical background

A pseudorandom number generator (PRNG) is a mathematical function that can generate a value that seems random [4]. PRNGs are initialized with an arbitrary random value called the seed. Values from a PRNG are reproducible if the seed is known. PRNGs are also characterized by their ability to produce many results in short periods of time [5].

This paper utilizes PRNGs to randomize pixels of a secret image, which is later imbedded in a cover image through steganography. The PRNGs used in this paper are: SplitMix, Well Equi-distributed Long-period Linear (WELL), Mersenne Twister, XORShift , Linear Congruential Generator (LCG) and Permuted Congruential Generator (PCG).

2.1. WELL

The WELL generator is a form of linear feedback shift register and a set of non-linear transformations. It is very similar to Mersenne Twister, yet exhibits longer periods and better equidistribution [6]. A general framework of how the WELL generator updates the state is described below.

$$Xi=(Xi-A \oplus (Xi-B \gg s\ ) \oplus (Xi-C \ll t)) \oplus T(Xi-D)$$

Xi: the current state
A, B, C, D*:* offset values which are predefined for each WELL variant.
Xi−A, Xi−B, Xi−C, Xi−D*:* the previous states at offsets A, B,C,D
T*:* Tampering function which includes other bitwise operations

2.2. SplitMix

The SplitMix generator is an object oriented and a splitable PRNG. While other PRNGs generate a value and update the state, this algorithm introduces the ability for an PRNG object to be split. In other words, an original PRNG object can be split into two pseudo-independent PRNG objects. The algorithm is especially useful in multithreaded programs, since the generation of pseudorandom numbers does not require synchronization [7]. The SplitMix algorithm is described below. The current state is updated by a fixed increment, where ***Y*** represents the next state. ***Y*** undergoes intermediate operations which are a series of right-shifting and multiplications.

$$Y= Xcurrent + \text{increment}$$

$$Y = Y \oplus (Y \gg 30) \times 0xbf58476d1ce4e5b9$$

$$Y=Y \oplus (Y \gg 27) \times 0x94d049bb133111eb$$

$$Y=Y \oplus (Y \gg 31)$$

2.3. *Mersenne* Twister

The Mersenne Twister is a PRNG known for its long period and effective randomness. It generates pseudo-random numbers by repeatedly executing a twisting operation, keeping an internal state vector, and starting with a seed value [8]. Twisting operation involves a series of bitwise, addition, and XOR operations to generates the next pseudo-random number in the sequence. It generates a long series of pseudo-random numbers with a period of $2^{19937} - 1$ before repeating.

$$Yi = Xi \oplus ((Xi >> \mathrm{u}) \& \mathrm{d}) \oplus ((Xi - \mathrm{m} \& \mathrm{a}))$$

Xi: the current state vector
Yi: the next pseudo-random number generated.
u, d, and a: constants used in the algorithm.
m: the length of the state vector.

2.4. XORShift

The XORShift algorithm generates pseudo-random numbers by performing bitwise XOR and shift operations to a seed value. It functions directly on a single integer number without requiring the maintenance of an explicit state vector and is renowned for its simplicity and efficiency [9]. Although XORShift has a shorter duration than some other PRNGs, it is well-liked for its efficiency and speed in a range of applications.

$$Xi+1= Xi \oplus (Xi \lll a) \oplus (Xi \ggg b)$$

Xi: the current state value
Xi+1: the next state value, which will be used to generate the next pseudo-random number
a and b: constants that determine the number of bits to shift.

2.5. LCG

An approach that uses a discontinuous piecewise linear equation to create a series of pseudo-randomized numbers is known as a linear congruential generator (LCG). One of the most well-known and ancient pseudorandom number generator algorithms is represented by the technique [10]. Their theory is not too complex, and they can be quickly and simply implemented, especially on computer hardware that supports modular arithmetic via truncation.

$$Xi+1= (a\ Xi+c) \bmod m$$

Xi: the current state value
a: the multiplier
c: the increment
m: the modulus
X_0: is the seed or starting value

2.6. PCG

A more recent method for producing pseudorandom numbers is the Permuted Congruential Generator (PCG), which aims to provide higher randomness and better statistical features [11]. To increase unpredictability, the PCG combines output permutation with linear congruential generation.

$$Xi+1=Xi\text{-}m+c \bmod 2^{l}$$

$$Yi = permute\ (Xi)$$

Xi: the current state
m and c : constants
2^l: the power of 2 defining the state size
Yi: the output after applying a permutation function to *X*i

3. Techniques and methods applied

In this section, the model design of the proposed secure communication system will be introduced. As illustrated in figure 1, image randomization and steganography will be applied by the sender before transmitting the message. Similarly, the reversed methods will be applied to the received image to retrieve the original secret image.

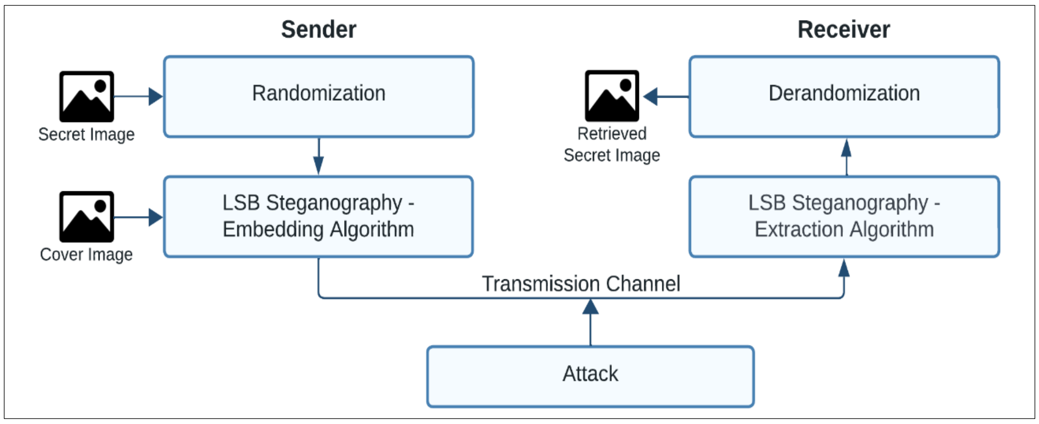


Figure 1 Overview of the scheme

First, the secret image shown in figure 2 will be randomized using 6 different randomization methods: SplitMix, WELL, Mersenne Twister, XORshift, LCG and PCG. Image randomization alters the pixel values in a random manner, making it more resistant to analysis. Afterwards, the randomized image will be hidden in the cover image using least significant bit (LSB) steganography. The embedding algorithm of LSB steganography is used to hide information, where each least significant bit of the cover image is sequentially replaced with a bit of the secret image.

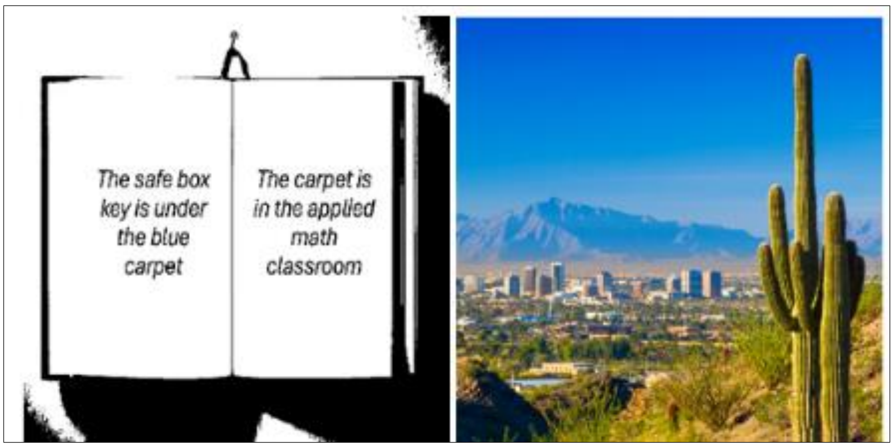

The safe box key is under the blue carpet
The carpet is in the applied math classroom

Figure 2 Secret image and cover image

During data transferring from the sender to the receiver through the transmission channel, some distortions or attacks may occur and modify the sent image. Therefore, three main distortions and attacks: noise addition, compression, and histogram attack, were applied on the transmitted image to test the robustness of the different randomization methods. Noise distortion was applied on the images using 20% and 50% strength rates, by adding random variations or disturbances into the pixel values. In addition, lossless compression was used, which reduces the size of the image without any loss of information. While noise addition and compression are caused by bad transmission, some attacks such as histogram attacks occur due to the analysis conducted by attackers.

The histogram attack works on identifying the most frequent pixel intensities (histogram peaks) and then averages the identified peaks with their adjacent pixel values. In this paper, the top 20 histogram peaks were targeted. This attack, when applied to a non-randomized secret message, was able to reveal some content of the secret message. However, the success of this attack greatly depends on the nature of the images.

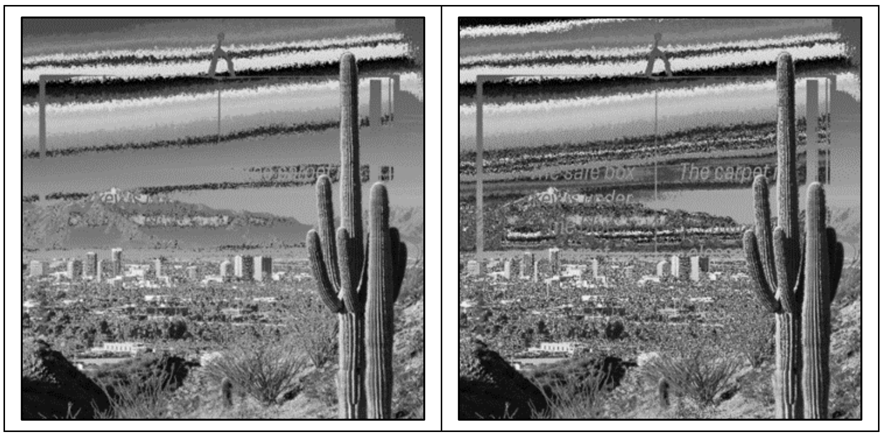


Figure 3 Stego image after histogram attack on 20 top peaks and 30 top peaks

At the receiver side, LSB steganography extraction algorithm and derandomization are done to retrieve the secret message. Finally, three different image comparison matrices will be used to quantify the quality of the retrieved images: MSE (Mean Squared Error), PSNR (Peak Signal-to-Noise Ratio), and SSIM (Structural Similarity Index). MSE determines the average squared difference between pixels in the original and reconstructed images. PSNR measures the quality of a reconstructed or compressed signal by determining the amount of noise in the reconstructed signal in comparison to the original signal. Furthermore, SSIM compares the similarity of two images based on luminance, contrast, and structure. It is frequently used to determine the perceived quality of image compression.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(I(i) - \hat{I}(i)\right)^2$$

$$\text{PSNR}=10\cdot \log_{10}(\text{MAX}^2/\text{MSE})$$

$$\text{SSIM}(x,y)=\frac{(2\mu_x\mu_y+C_1)(2\sigma_{xy}+C_2)}{(\mu_x^2+\mu_y^2+C_1)(\sigma_x^2+\sigma_y^2+C_2)}$$

N: the total number of pixels in the image.
I(i), Î(i): the pixel values of the original and reconstructed images, respectively.
μ_x, μ_y: the means of images x and y respectively.
σ_x^2, σ_y^2: the variances of images x and y respectively.
σ_{xy}: the covariance of images x and y.
C_1, C_2: constants to stabilize the division with weak denominator.

4. Experimental results

The randomized secret image was hidden in the cover image in figure 2 using LSB steganography. Then, it went through three different distortions, and the performance of each randomization method was measured. The performance is divided into two parts, numerical part using MSE, PSNR, and SSIM, which analyses the performance according to pixels, and the second part is visual, which analyses the readability of the retrieved image.

4.1. Numerical Results

Table 1 Retrieved image assessment after 20% noise addition

Metric/ Scheme	No Randomization	SplitMix	WELL	Mersenne Twister	XORshift	LCG	PCG
MSE	0.28753	0.28811	105.07	0.28722	0.28706	0.28741	0.28762
PSNR	53.543	53.535	27.915	53.549	53.551	53.5457	53.5425
SSIM	0.10137	0.1012	0.068684	0.10185	0.10135	0.10128	0.10150

First, the stego-image went through 20% noise addition, and after retrieving the image, it was compared with the original secret image before randomization. Table 1 shows the results of the 20% noise addition with six randomization methods and without randomization.

From the table below, WELL has the highest MSE, whereas the others, along with no randomization, have very close results of 28% error. Hence, randomization with SplitMix, Mersennse Twister, XORshift, LCG, and PCG, retrieve the image as without randomization (numerically).

Secondly, the stego-image went through 50% noise addition to see how increasing the noise can affect the performance of the randomization methods. Results are illustrated in table 2.

Table 2 Retrieved image assessment after 50% noise addition

Metric/ Scheme	No Randomization	SplitMix	WELL	Mersenne Twister	XORshift	LCG	PCG
MSE	0.46650	0.4674	105.22	0.45889	0.45875	0.46012	0.46008
PSNR	51.442	51.433	27.909	51.514	51.515	51.502	51.502
SSIM	0.052954	0.05229	0.04231	0.05286	0.05307	0.05268	0.05298

From the results above, WELL is also performing the worst in terms of image retrieval, while other randomization methods have around 46% error, pixel-wise, where this percentage may not reflect visually.

Thirdly, the stego-image went through compression, as it is very common for images to get compressed when transmitted. Table 3 below shows the numerical results for the PRNGs.

Table 3 Retrieved image assessment after compression

Metric/ Scheme	No Randomization	SplitMix	WELL	Mersenne Twister	XORshift	LCG	PCG
MSE	0.17310	0.17310	105.02	0.17311	0.74452	0.17310	0.17310
PSNR	55.747	55.747	27.918	55.748	49.412	55.7476	55.747
SSIM	0.99934	0.9993	0.09219	0.99934	0.00174	0.99934	0.99934

According to table 3 above, the image was restored without randomization, as the SSIM is 0.999 that is very close to 1, which is due to using the "PNG" format in the image. Also, the image was restored successfully in SplitMix, Mersenne Twister, LCG, and PCG, whereas WELL and XORshift failed to retrieve the image after it got compressed, as their SSIM is closer to 0.

Finally, the stego-image was exposed to a histogram attack to assess the performance of the PRNGs against malicious attacks. Results are illustrated in table 4.

Table 4 Retrieved image assessment after histogram attack

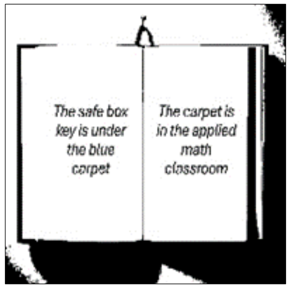
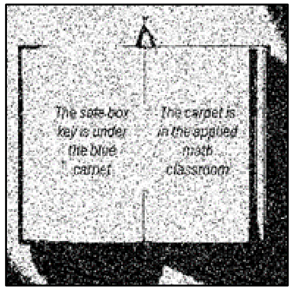
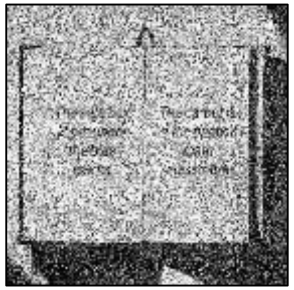
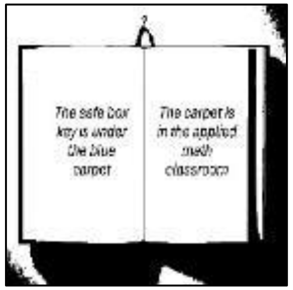
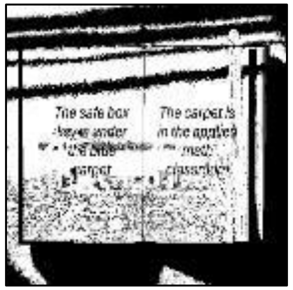
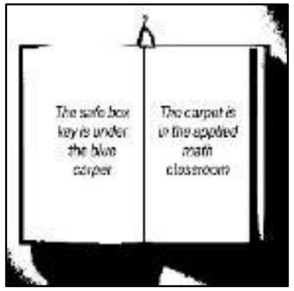
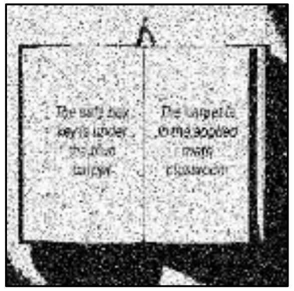
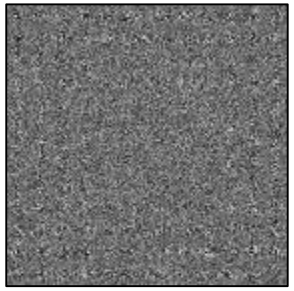
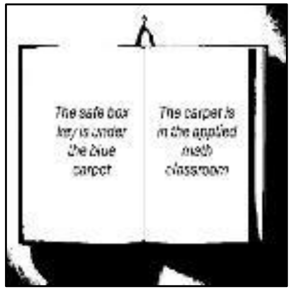
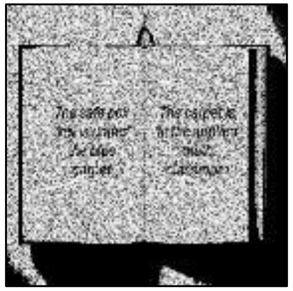
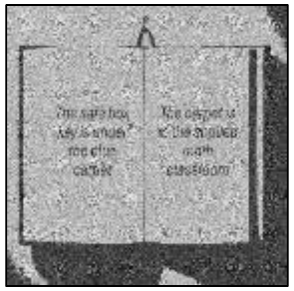
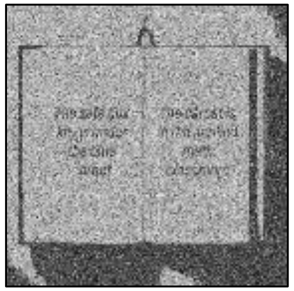
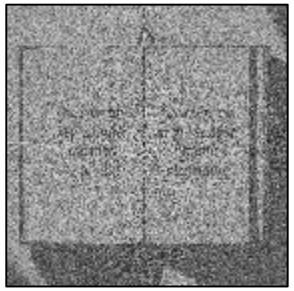
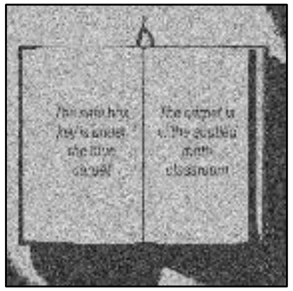
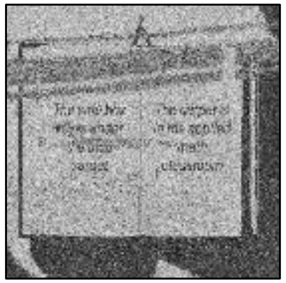
Metric/ Scheme	No Randomization	SplitMix	WELL	Mersenne Twister	XORshift	LCG	PCG
MSE	0.37686	0.38838	105.10	0.74688	0.29918	0.34993	0.38004
PSNR	52.3689	52.238	27.914	49.398	53.372	52.6908	52.332
SSIM	0.56501	0.28588	0.076652	0.00180	0.51939	0.29240	0.28663

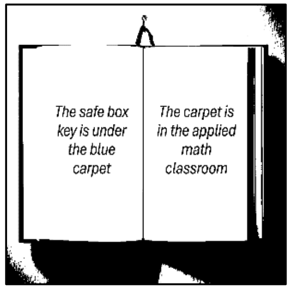
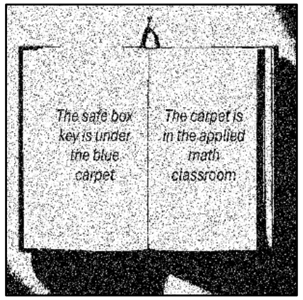
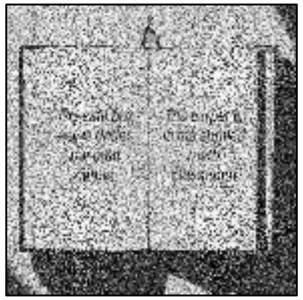
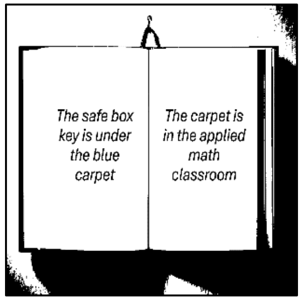
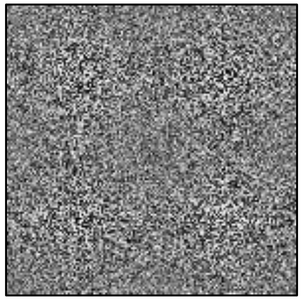
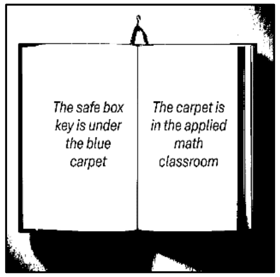
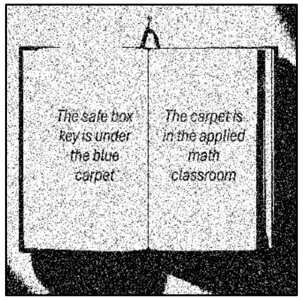
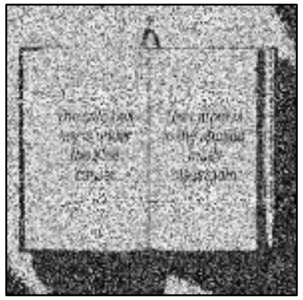
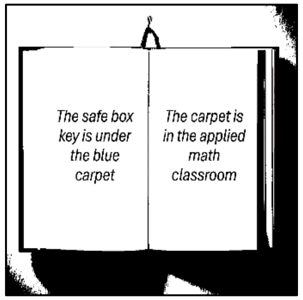
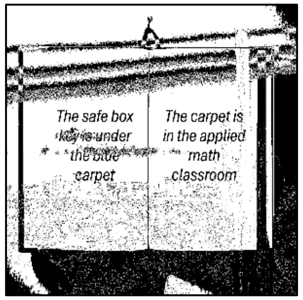
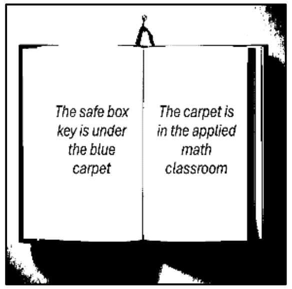
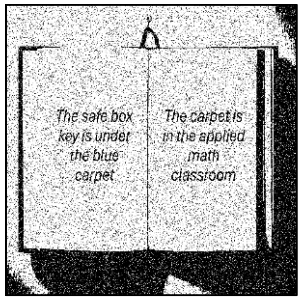
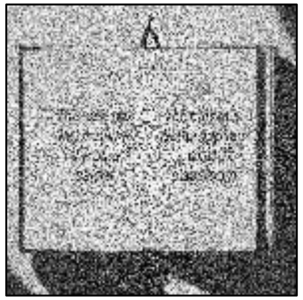
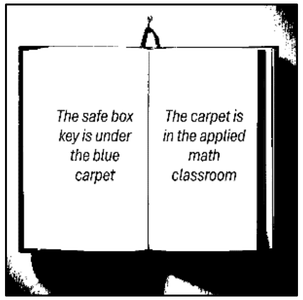
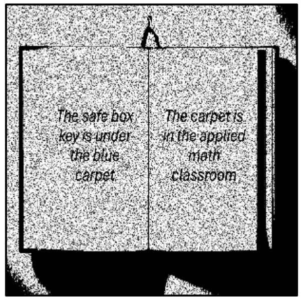
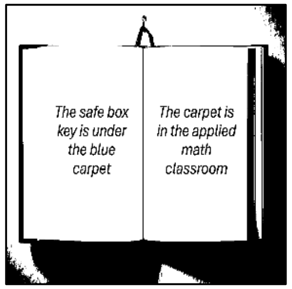
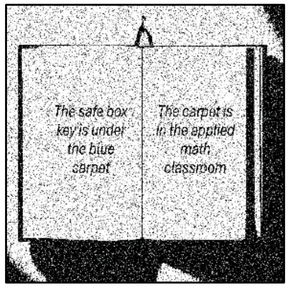
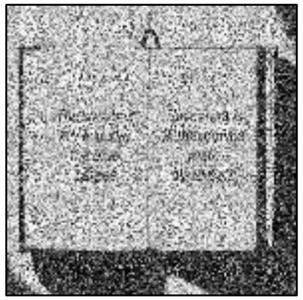
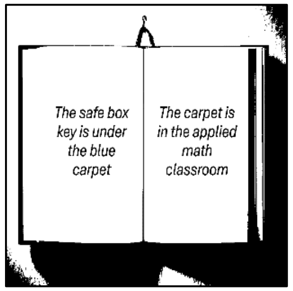
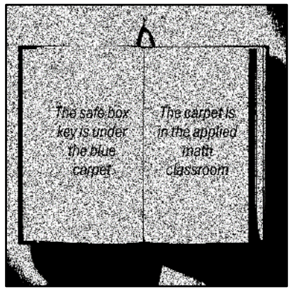
According to the results displayed in table 4, WELL and Mersenne Twister have the worst performance among all, where the SSIM is almost 0, and the MSE is very high. As for the others, they have almost the same error percentage close to 35%, where no randomization and XORshift show the least noise, as their SSIM is around 0.5.

4.2. Visual Results

The visual results for each attack with each randomization technique are illustrated in table 5, which will be verifying the numerical results obtained.

Table 5 Visual results for image retrieval

Metric/ Scheme	**No Attack**	**20% Noise Addition**	**50% Noise Addition**	**Compression**	**Histogram Attack**
No Randomization	 The safe box key is under the blue carpet The carpet is in the applied math classroom				
SplitMix	 The safe box key is under the blue carpet The carpet is in the applied math classroom				
WELL					

Mersenne Twister					
XORshift					
LCG					
PCG					

From table 5, it is clear that without randomization, the PNG image is restored well with compression and with 20% noise, whereas with the histogram attack, the image was not restored, and it has a lot of distortion. Generally, LCG, PCG, and XORshift have the best performance, where under all conditions, the secret image was successfully retrieved, and the text can be extracted from the images. For WELL, it has a bad performance among all attacks, whereas twister fails completely with the histogram attack. Finally, SplitMix is affected badly by increased addition of noise, and it stops retrieving the secret image.

5. Conclusions

With the challenges that affect steganography, PRNGs were used to prevent attacks and distortions from wiping away the secret image information from the stego-image. Six PRNGs were used, SplitMix, WELL, Mersenne Twister, XORshift, LCG, and PCG. From the results, it was concluded that PNG Images, without randomization, are resilient against distortions caused by transmission in the network. Additionally, without randomization, it is hard to retrieve the secret image when the stego-image is attacked. Finally, LCG, PCG, and XORshift are resilient against both distortions and attacks.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Kaspersky (2023) *What is steganography? definition and explanation*, *www.kaspersky.com*. Available at: https://www.kaspersky.com/resource-center/definitions/what-is-steganography (Accessed: 03 May 2024).

[2] A. A. Abdulla, Digital Image Steganography: Challenges, investigation, and recommendation for the future direction," *Soft Computing*, Sep. 2023. doi:10.1007/s00500-023-09130-8.

[3] GeeksforGeeks, Pseudo random number generator (PRNG), GeeksforGeeks, https://www.geeksforgeeks.org/pseudo-random-number-generator-prng/ (accessed May 3, 2024).

[4] V. Tilborg, Encyclopedia of cryptography and security. New York: Springer, 2011.

[5] Tom St Denis and S. Johnson, Cryptography for developers. Rockland, Ma: Syngress Publishing, Inc, 2007.

[6] P. L'Ecuyer and F. Panneton, "Improved Long-Period Generators Based on Linear Recurrences Modulo 2," Les Cahiers du GERAD, pp. 1–17, Aug. 2004.

[7] G. L. Steele, D. Lea, and C. H. Flood, Fast splittable pseudorandom number generators," SIGPLAN notices, vol. 49, no. 10, pp. 453–472, Oct. 2014, doi: https://doi.org/10.1145/2714064.2660195.

[8] Matsumoto, M. and Nishimura, T. (1998) 'Mersenne twister', ACM Transactions on Modeling and Computer Simulation, 8(1), pp. 3–30. doi:10.1145/272991.272995.

[9] Panneton, F. and L'Ecuyer, P. (2005) 'On the XORshift random number generators', ACM Transactions on Modeling and Computer Simulation, 15(4), pp. 346–361. doi:10.1145/1113316.1113319.

[10] Linear Congruential Generator, Wikipedia, https://en.wikipedia.org/wiki/Linear congruential_generator#:~:text=A%20linear%20congruential%20generator%20(LCG,known%20pseudorandom%20number%20generator%20algorithms. (accessed May 3, 2024).

[11] Permuted Congruential Generator, Wikipedia, https://en.wikipedia.org/wiki/Permuted congruential_generator#:~:text=A%20permuted%20congruential%20generator%20(PCG,2n%20linear%20congruential%20generator. (accessed May 3, 2024).