

Article

EPRNG: Effective Pseudo-Random Number Generator on the Internet of Vehicles Using Deep Convolution Generative Adversarial Network

Chenyang Fei ¹, Xiaomei Zhang ^{1,*} , Dayu Wang ² , Haomin Hu ¹, Rong Huang ¹ and Zejie Wang ¹

¹ School of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai 201620, China; qq1697809501@163.com (C.F.); hhm-hao@163.com (H.H.); huangrong62@163.com (R.H.); zejiewang@126.com (Z.W.)

² College of Health Science and Environmental Engineering, Shenzhen Technology University, Shenzhen 518118, China; 2210417013@email.szu.edu.cn

* Correspondence: xmzhang@sues.edu.cn; Tel.: +86-21-6779-1035

Abstract: With the increasing connectivity and automation on the Internet of Vehicles, safety, security, and privacy have become stringent challenges. In the last decade, several cryptography-based protocols have been proposed as intuitive solutions to protect vehicles from information leakage and intrusions. Before generating the encryption keys, a random number generator (RNG) plays an important component in cybersecurity. Several deep learning-based RNGs have been deployed to train the initial value and generate pseudo-random numbers. However, interference from actual unpredictable driving environments renders the system unreliable for its low-randomness outputs. Furthermore, dynamics in the training process make these methods subject to training instability and pattern collapse by overfitting. In this paper, we propose an Effective Pseudo-Random Number Generator (EPRNG) which exploits a deep convolution generative adversarial network (DCGAN)-based approach using our processed vehicle datasets and entropy-driven stopping method-based training processes for the generation of pseudo-random numbers. Our model starts from the vehicle data source to stitch images and add noise to enhance the entropy of the images and then inputs them into our network. In addition, we design an entropy-driven stopping method that enables our model training to stop at the optimal epoch so as to prevent overfitting. The results of the evaluation indicate that our entropy-driven stopping method can effectively generate pseudo-random numbers in a DCGAN. Our numerical experiments on famous test suites (NIST, ENT) demonstrate the effectiveness of the developed approach in high-quality random number generation for the IoV. Furthermore, the PRNGs are successfully applied to image encryption, and the performance metrics of the encryption are close to ideal values.

Keywords: pseudo-random number; DCGAN; IoV; early stopping



Academic Editor: Vasco N. G. J. Soares

Received: 3 December 2024

Revised: 28 December 2024

Accepted: 29 December 2024

Published: 3 January 2025

Citation: Fei, C.; Zhang, X.; Wang, D.; Hu, H.; Huang, R.; Wang, Z. EPRNG: Effective Pseudo-Random Number Generator on the Internet of Vehicles Using Deep Convolution Generative Adversarial Network.

Information **2025**, *16*, 21. <https://doi.org/10.3390/info16010021>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Currently, the Internet of Vehicles (IoV) has been an indispensable part of using smart vehicles and protecting traffic safety. To continuously transmit and share information, vehicles communicate with each other, with roadside units (RSUs), and with personal mobile devices through wireless technology on the IoV. A wide variety in data transmission systems bring intelligence to modern transportation and convenience to our daily life [1]. However, open, shared, wireless mediums on the IoV create many security and privacy threats, which have attracted growing attention from academics and the industry. Therefore,

how to effectively protect private data and vehicular communications against threats imposed by attacks and unauthorized access is a problem of paramount importance on the IoV [2].

American IEEE 1609.2 [3] and European ETSI ITS-G5 [4] show that cryptography has been an important foundation to guarantee the security and privacy of vehicular network communications. With the purpose of an encryption key with good randomness, it is an essential task to design highly secure random number generators (RNGs) [5]. Random number generators (RNGs) can be categorized into true random generators (TRNGs) and pseudo-random number generators (PRNGs). Traditional TRNGs are based on physical phenomena, such as various types of field programmable gate arrays (FPGA) [6], thermal noise [7], and biological signals [8] to produce random numbers. Unpredictability from physical phenomena is cited as one of the main reasons that high-security applications usually adopt TRNGs. In fact, collecting physical phenomena and the need for specific hardware make TRNGs inconvenient for the required long generation time. To tackle these challenges, researchers have proposed pseudo-random number generators (PRNGs) as another choice for ordinary purposes. The architectures of PRNGs are able to rapidly generate sufficiently complex and random numbers through initial values and algorithms [9]. PRNGs are preferred for their practicability in real scenarios.

Though extensive efforts have been devoted to PRNGs, open and public PRNG algorithms bring corresponding challenges to vehicle applications [10]. Traditional PRNGs generate the same random numbers with a period if we use the same inputs. Therefore, seeds must be hidden carefully in order to avoid malicious users deducing the number generation pattern [11]. It becomes quite difficult and expensive to completely hide the PRNG seeds. Considering that algorithms may be public, an unpredictable and low-cost PRNG is greatly desired. Deep learning-based PRNGs generators have been proposed to create PRNGs [12]. In existing solutions, mathematical functions are approximated to generate new PRNGs without explicit instructions. However, due to the real interference in driving scenarios and dynamics in training processes, there exist low-random outputs and a pattern collapse, and there is still a need for further effort to improve the system reliability for safety applications. In this paper, we propose a DCGAN approach [13] to generate PRNGs for the IoV that takes into consideration both the interference from environments and dynamics of training processes. Our method can both continuously generate chaotic sequences with high randomness and avoid overfitting during training processes of networks. Moreover, the periodicity of our EPRNG can be eliminated by upgrading training sets.

Smart vehicles are typically equipped with built-in sensors that can capture road information for intelligent driving decisions [14]. Although existing deep learning solutions utilize the rich data of a device to train and obtain random and different results, their application for PRNGs using raw vehicle data brings new challenges. In fact, the generator may receive interference from actual unpredictable driving environments, for example, duplicate images captured by vehicles camera from roads or low-quality images in extreme darkness or brightness. This means the entropy values of such images are low, which may further lead to low-randomness outputs. We design an image stitching method for extracting high-entropy pixel blocks to regenerate the combined images, which reduces the duplication while minimizing the loss of image information. Meanwhile, data collected from other sensors on the vehicles are added into images as noises and compensate for the loss of photo quality caused by extreme environments. These are intended to achieve the purpose of image entropy enhancement initially, thereby increasing the randomness and unpredictability of the data, then input these data into our DCGAN to improve the randomness of the generation model. Accordingly, our DCGAN image outputs with high

entropy are leveraged as inputs of the chaotic algorithm [15] to produce pseudo-random number sequences that can pass statistical test suites.

Recently, GAN-based PRNGs which dynamically generate one-time sequence in the game between generator and discriminator have been exploited to generate high-quality pseudo-random numbers. However, they suffer from training instability and pattern collapse in GANs [16]. The training epochs of high-entropy image generation models may be unable to be accurately located, which causes overfitting and wastes computing power. Meanwhile, the data entropy plays an important role in reflecting the randomness of data [17], the higher the randomness and unpredictability of the data, the higher the entropy value. Therefore, we introduce image entropy to monitor the training process of the DCGAN. We present an entropy-driven stopping method that stops the training process when the average value and the variance of the image entropy reach predefined thresholds. On the one hand, we can automatically locate the convergence point of training for high-entropy images, which optimizes the training sessions and conserves computing power. On the other hand, our method can prevent overfitting since the model automatically stops training when a convergence point is detected.

The main contributions of this paper are summarized as follows:

1. We present a DCGAN-based pseudo-random number generation for the IoV, which is expected to achieve high randomness and save resources, based on extracting entropy solely from vehicle sensor data.
2. We present a data preprocessing method including image stitching and noise adding, which extracts high-entropy pixel blocks and regenerates the combines images from duplicate images and then adds data from other vehicle sensors as image noises. This way, the influence of environmental interference on images (inputs of DCGAN) is reduced, and the randomness of the results can be improved.
3. The entropy-driven stopping method aims to establish generic automatic localization of model training's convergence points, applied to image-based GAN models. The novelty of this method mainly lies in the use of image entropy for monitoring and stopping the training process to avoid overfitting and conserve computing power. Additionally, we define a novel index, the effectiveness score, that quantitatively evaluates the performance of the proposed stopping method in our experiments.
4. We evaluate the effectiveness of our PRNG on test suites and show that it can both pass approximately 98% of NIST test suites [18] and all ENT test suites [19] and prevent overfitting, indicating the completely uniform probability distribution and its high statistical randomness. Our experimental results in image encryption quantitatively show that the encryption's Unified Average Changing Intensity (UACI) and Number of Pixels Change Rate (NPCR) are successfully passed at the 0.05 significance level and demonstrate excellent resistance against different attack scenarios.

The rest of this article is organized as follows. Section 2 introduces related work about existing RNGs in recent years. Section 3 presents the construction of the proposed PRNG. Section 4 uses some simulation methods to analyze the effectiveness and the security of the proposed PRNG. Section 5 summarizes this article.

2. Related Work

RNGs have been intensively investigated recently to generate encryption keys for protecting data security. RNGs can be divided into TRNGs and PRNGs. TRNGs usually use physical phenomena as entropy sources to generate true random numbers, including FPGA [6], sensor-based IoT [20], and quantum random number generator [21], while PRNGs process random input seeds to produce large number sequences through mathematical functions [22,23].

2.1. Image as RNGs Inputs

Most traditional RNGs usually rely on specialized and costly equipment, demanding significant manpower and material resources. Therefore, reducing the cost of entropy collection has become a critical challenge for RNGs. To address these issues, some researchers have proposed using images captured by mobile phones as an entropy source for generating random numbers. However, images taken from similar scenes tend to exhibit shared features, which can introduce correlations in the resulting random numbers. L. Zhao et al. [24] employed chaos-based cryptographic techniques to process the captured images, effectively generating high-quality true random numbers. This highlighted the potential of images as a reliable entropy source. W. Z. Yeoh et al. [25] introduced parallelization and other optimization method to enhance the efficiency of chaos-based algorithms for true random number generation. Their approach achieved a throughput of approximately 118.9 Mbps, demonstrating that images could not only serve as an entropy source but also enable the generation of high-quality true random numbers without the need for additional external hardware. However, the aforementioned methods applied images captured in relatively ideal environments without considering the effect of light and movement.

In support of the goal of producing high-quality random numbers, we also explore pictures captured while driving as training sets. To address the challenges posed by extreme environments and unpredictable movements, we adopt a specialized preprocessing method that extracts entropy sources from similar images and integrates data collected from other vehicle sensors as noise sources of images. This approach can reduce environmental interference in extreme conditions and then effectively enhance the entropy of collected data, thereby mitigating the instability of the quality of data from vehicle sensors.

2.2. Machine Learning in RNGs

Unlike TRNGs, PRNGs simply rely on software to generate high-quality pseudo-random numbers without the necessity for external hardware or third-party services. However, the deterministic nature of the inputs can introduce vulnerabilities, posing significant security risks. With the development of machine learning (ML), researchers have proposed leveraging ML techniques to strengthen PRNG security. N. D. Truong et al. [26] introduced a predictive ML analysis to reinforce the security of various stages in quantum random number generators (QRNGs). Similarly, L. Pasqualini et al. [12] applied Reinforcement Learning (RL) to solve a partially observable Markov Decision Process (MDP), enhancing the security of PRNGs. Although these ML-based methods effectively improve the security of PRNGs, they often rely on public chaotic algorithms as training datasets and still face significant security risks due to potential leaks of algorithms and inputs. We utilize updated data collected from vehicle sensors as our training dataset. They offer the following features: (i) no additional data acquisition, (ii) dataset updated in real time, (iii) variability in roadside data, and (iv) high-dimensional property. Therefore, it is suitable for complex deep learning and can defend against leaks of fixed algorithms and inputs.

Among these ML technologies, PRNGs based on GANs have also received some attention in recent years. GANs are unsupervised deep learning neural network model architectures capable of generating realistic and diverse data from a specified distribution. M. De Bernardi et al. [27] pioneered the use of GAN for generating pseudorandom numbers, designing a method where the generator's output was concealed, and the discriminator was trained to learn a mapping from observable to concealed data. This approach successfully passed statistical test suites, such as NIST 800-22, demonstrating its high security and reliability for generating pseudo-random numbers. Building on this, P. Ji et al. [28] utilized one-dimensional data as training sets, generating pseudo-random numbers that exhibited

dynamic properties, enhancing the unpredictability of GAN-based PRNGs. Furthermore, variations of GANs have been explored to improve PRNG performance, and these methods also pass the statistical test suites. For instance, K. Okada et al. [11] demonstrated that a Waassertein distance-based generative adversarial network (WGAN) could generate high-quality pseudo-random numbers using predictable data as inputs. That study showed that GANs could effectively learn the randomness from the training sets, independent of the randomness in the input data. Z. Man et al. [29] proposed the use of a least-squares GAN to generate random numbers, which were then utilized to encrypt images, showing effectiveness through NPCR and UACI tests [30]. Moreover, alternative training sets have also been introduced to further improve PRNG security. For example, J. Li et al. [31] used quantum random numbers as GAN training data, enhancing the security of GAN-based PRNGs by incorporating the unpredictability of quantum randomness. However, resources are wasted via training instability and pattern collapse in GANs. In this paper, we design an entropy-driven stopping method, inspired by previous researchers [32] which applied early-stopping methods in deep learning. Addressing the goal of maximizing output randomness, we stop our model training at the optimal epoch so as to prevent overfitting and establish appropriate evaluation criteria that ensure the effectiveness and randomness of the generated outputs.

2.3. RNGs for IoV

With the rapid advancement of sensing capabilities in mobile devices, the IoT faces increased risks of intrusion, thereby increasing the need for high-quality random numbers. IoT-based RNGs can utilize the extensive autonomous sensor data generated by mobile devices as inputs to chaos algorithms for random number generation. K. Wallace et al. [16] suggested that high-quality sensors with efficient power management in IoT systems could significantly enhance the performance of TRNGs in practical applications. Additionally, Q. Shi et al. [33] used QRNGs as an entropy source to design an authentication scheme that could protect the security of cryptography in the IoV. Unlike a traditional authentication scheme, QRNG-based RNGs use true random numbers as seeds for parameter generation, enabling secure access management and enhancing IoV data security.

In this paper, we propose a method using deep learning to enrich the entropy source collected by vehicle sensors for PRNGs on the IoV, which utilizes the rich data of vehicles. To the best of our knowledge, this is the first work that studies deep learning-based PRNGs for the IoV. By leveraging the powerful data processing capabilities of deep learning, our method transforms vehicle sensor data into high-quality random numbers, thereby introducing intelligence and enhancing the security of modern transportation systems. A detailed analysis and comparison of existing RNG methods is presented in Table 1. Compared to the other three PRNGs in [11,24,27], our EPRNG can achieve strong randomness and robustness, because these three methods utilize pseudo-random numbers as their training sets, while our EPRNG utilizes physical information as its training set. Moreover, an entropy-stopping method is designed to regulate the training process of our EPRNG, which can effectively mitigate overfitting and then enhance the robustness of PRNG during operation.

Table 1. Existing RNG: analysis and comparison (supported: ✓, unsupported: ✕).

Related Work	Statistical Test Suites (NIST or ENT)	Rely on Third Part	Robustness	Hardware /Software	Entropy Source /Training Sets	Advantages	Limitations
[6]	✓	✕	-	Ring oscillator in general	Electron noise	• Unpredictable and true random entropy source	• Electronic circuits can age and lead to poor performance
[7]	✓	✕	-	130 nm CMOS technology in general	Electronic noise	• Unpredictable and true random entropy source • Simple collecting entropy source methods	• The randomness of entropy source is affected by circuit aging
[33]	✕	✓	-	QRNG	QRNG	• Unpredictable and true random entropy source • Resistance to traditional attacks	• Professional and expensive equipment • Difficult to implement protocol
[24]	✓	✕	-	Mobile phone camera	Image	• Unpredictable and true random entropy source • Low-cost device	• Users must keep monitoring their mobile phone to take pictures
[27]	✓	✓	✕	Software	Pseudo-random numbers	• Almost unpredictable • Low-cost and convenient data collection	• Predictable training sets • The training of GAN is out of control
[11]	✓	✓	✕	Software	Pseudo-random numbers	• Almost unpredictable • Low-cost and convenient data collection • With predictable data as inputs	• Predictable training sets • The training of GAN is out of control
[34]	✓	✕	✕	Embedded processors	Pseudo-random numbers	• Almost unpredictable • High generating speed	• Predictable training sets • The training of GAN is out of control • Requires external professional hardware
Our proposed method	✓	✕	✓	Software	Pseudo-random numbers	• Almost unpredictable • Low-cost and convenient data collection • With predictable data as inputs	• Predictable training sets • The training of GAN is out of control

3. PRNG System Design

3.1. System Architecture

In this section, we present the system architecture of the proposed PRNG for the IoV. The architecture is shown in Figure 1, including the local vehicles, roadside units (RSU), and fog server. We use road condition images as primary datasets and other sensor data as ancillary datasets; the data can be captured by vehicle sensors from the outside environment. If the local vehicles lack sufficient computing power to process and train these data, fog computing can be utilized to generate pseudo-random numbers. Furthermore, in order to reduce the burden of transmission, the vehicles send the collected data to the nearby RSUs, which apply the proposed preprocessing method to generate high-entropy images as training sets for DCGANs before transmitting them to the fog server. The fog server then generates pseudo-random numbers and transmits them back to the vehicles.

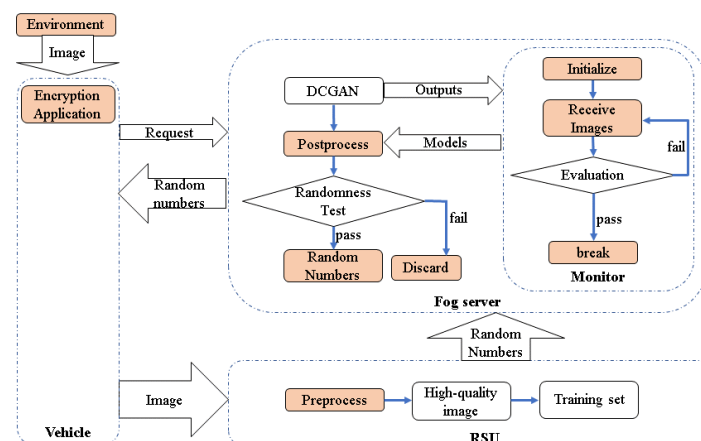


Figure 1. System model of the proposed PRNG.

The network architecture, shown in Figure 2, plays a crucial role in enabling the interaction between vehicles, RSUs, and the fog server. The communication network supports various modes, including vehicle-to-vehicle (V2V), vehicle-to-RSU (V2R), vehicle-to-fog server (V2F), and RSU-to-fog server (R2F). Each vehicle is equipped with cameras and sensors to collect road condition data. When a vehicle cannot process these images locally, it transmits them to the nearest RSU using the V2R communication mode.

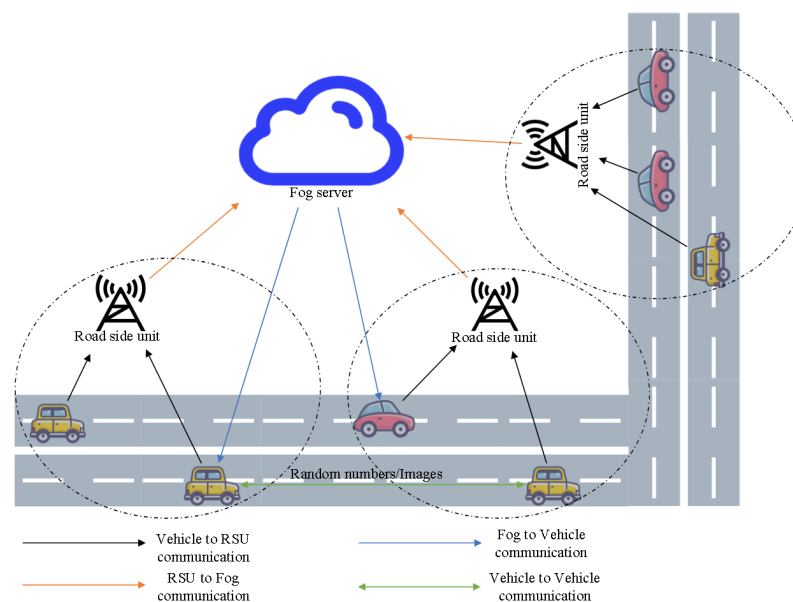


Figure 2. Network model of the IoV.

Upon receiving the data, RSUs preprocess the data using the proposed method described in Section 3.2 to enhance the entropy of the collected data. These high-entropy images are then transmitted via the R2F link to the fog server. The fog server employs DCGAN to train the preprocessed images, generating high-entropy output images. This process is reinforced by the proposed entropy-driven stopping method to enhance the robustness of the DCGAN. After training, the fog server applies a postprocessing method based on chaotic logic algorithms to extract pseudo-random numbers from the generated images. Once the pseudo-random numbers are generated, they undergo validation through statistical test suites and encryption simulations to ensure uniform distribution, independence, and randomness. Vehicles can request these pseudo-random numbers from the fog server via the V2F communication link and use them to secure their communications and data within the IoV system.

The use of fog computing, initially introduced by Bonomi et al. [35], can support computing and storage resources for vehicles on the IoV [36]. Moreover, edge devices are usually located close to the vehicles. Therefore, the latency of information transmission is very low, and it can offer real-time connection between the vehicles and the fog server.

In summary, the proposed PRNG system integrates both system-level processing (preprocessing in RSUs and training in the fog server) and network-level communication (V2R, R2F, and V2F), ensuring that the generated pseudo-random numbers meet the required security standards while minimizing the computational burden on individual vehicles.

3.2. Data Preprocessing

The datasets used in this article were proposed by KITTI [37]. The datasets contain a number of data collected by vehicles sensors. We used the images as primary datasets, but there were problems of data duplication and low entropy, so the data needed to be preprocessed before adding them to the training sets.

3.2.1. Image Stitching

Images are randomly captured by the vehicle cameras while driving on the road, and the vehicle cameras may receive interference from duplicate images. Figure 3 illustrates the correlation among the images. The x -axis and y -axis represent randomly selected images, where darker colors indicate higher similarity between the images. This suggests that a significant correlation exists among the collected images, so corresponding measures must be taken to reduce the duplication while minimizing the loss of image information. In this paper, we designed an image stitching algorithm to enhance the entropy of the images, especially the sets with low entropy. From Figure 4 which compares the image entropy before and after stitching, it is evident that the image stitching can not only increase the image entropy but also achieve the target image entropy (≥ 7.5). Image entropy measures the unpredictability and randomness of an image. The method consists of the following main stages:

- (1) Before the image is stitched, the images need to be cropped to a uniform size, and then we use the smallest size in the sets of images as the target size for all images.
- (2) Set the parameters of the blocks and stitched images. Table 2 shows the parameters we set.

Table 2. The parameters of image stitching.

Size of Block	Size of Stitched Image	Target Entropy
10×10	Data 1000×500	7.5

- (3) Segment the captured images, and calculate their values of entropy by Equation (1).

$$H(X) = - \sum_{i=1}^n p_i \log_2 p_i \quad (1)$$

where $H(x)$ represents the image entropy of an image X , p_i represents the probability of the i th pixel value in the image X . In the case of an 8-bit image, where each pixel value can range from 0 to 255, the maximum entropy value is 8 bits. This maximum value occurs when all possible pixel values are equally likely, meaning $p_i = \frac{1}{256}$.

- (4) Arrange the blocks in ascending order based on their entropy values and name blocks as $a_0, a_1, a_2, \dots, a_n$; the result is shown in Figure 5.
- (5) If the entropy value of the combined block reaches the target entropy value, retain the combined block as block b . Then, combine block b with the next pixel block in the sequence.
- (6) Use Algorithm 1 to obtain the stitched image.

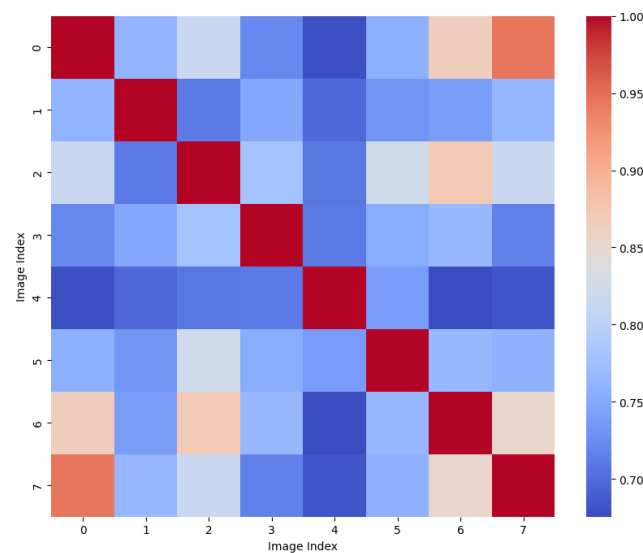


Figure 3. Similarity between images.

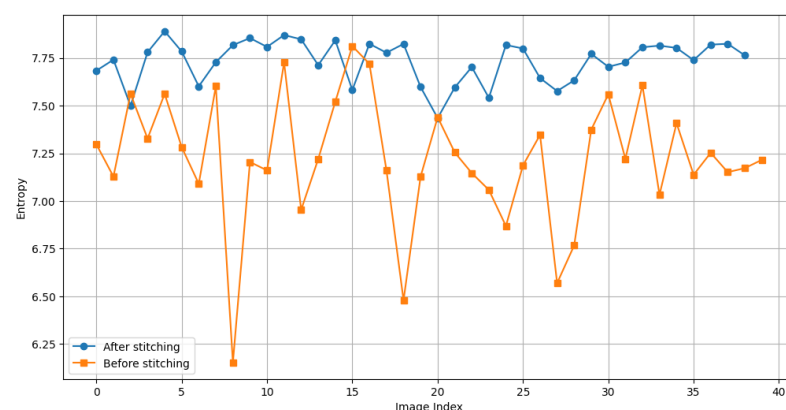


Figure 4. Comparison of images before and after stitching.

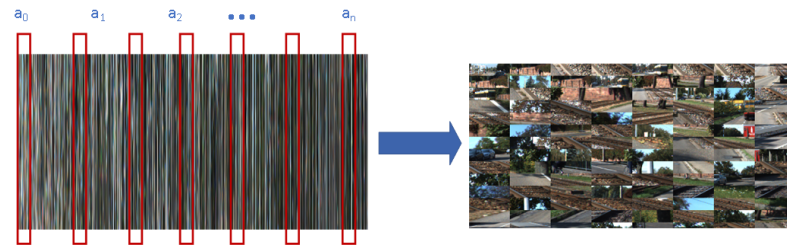


Figure 5. Stitching the pixel blocks.

Algorithm 1: Algorithm for image stitching

Input: images x , target_entropy T
Output: merage_blocks y , output_image o

```

1 pixel_blocks ← segmentation( $x$ );
2 for pixel_blocks to images_blocks do
3   //Calculate the entropy of all image blocks;
4   image_blocks ← calculate_entropy(pixel_blocks);
5   //Sort by entropy size  $a \leftarrow \text{sorted}(\text{image\_blocks})$ ;
6 end
7 //Calculate the highest entropy of the blocks;
8 new_entropy ← Entropy( $a[0]$ );
9 for  $i = 0$  to max_num do
10  entropy ← Entropy( $a[i]$ );
11  //Determine whether the target entropy value is reached;
12  if entropy  $\geq T$  and new_entropy  $\geq T$  then
13    //merage blocks;
14     $b \leftarrow \text{merage}(a[i], a[i + 1])$ ;
15    new_entropy ← Entropy( $b$ );
16    entropy ← new_entropy;
17    Update the  $y$ ;
18  else
19    continue
20  end
21 end
22 return  $y$ ;
23  $o \leftarrow \text{Transform}(y)$ ;

```

3.2.2. Noise Adding

Because the motion of the vehicle is usually unpredictable, the vehicle camera may receive interference from images in a hard environment (extreme dark or extreme light). In this paper, noise sources collected from the other vehicle sensors in the KIITI datasets can be added to the images, including the Inertial Measurement Unit (IMU) for the state of motion of the vehicle (acceleration, velocity, and angular rate), the global positioning system (GPS) for precision location (latitude and longitude), and the operator of GPS and IMU sensors for the parameter setting of the IMU and GPS.

Furthermore, to evaluate the performance of the noise-adding algorithm under extreme conditions, we reduced the image brightness to 30% of its original level to simulate a harsh environment, and the simulation is shown in Figure 6. We then compared the entropy changes before and after adding noise. The results are shown in Figure 7, where the x -axis represents randomly selected images, and the y -axis is their entropy value. Figure 7

demonstrates that the proposed noise-adding method can effectively enhance the entropy in harsh environments. The method can be broken down into the following steps:

- (1) Flatten the random data into a one-dimensional array.
- (2) Use the noise source to construct a noise matrix, whose dimensions are consistent with the image.

$$N(i, j) = \text{random_data}[i, j], \quad 0 \leq i < H, 0 \leq j < W \quad (2)$$

where i and j represent the pixel indices of the noise, H represents the width of the image, and W represents the height of the image.

- (3) In order to avoid structural patterns in the noise, we shuffle each pixel of the noise to ensure the noise is random. Figure 8 shows 16 randomly selected data samples from the datasets. The data are normalized into gray-scale blocks within the range of (0, 255), with the x -axis and y -axis representing the data indices.
- (4) In order to simulate the intensity of inhomogeneity in the real world and enhance the randomness as much as possible, we use Equation (3) to set the noise intensity.

$$\text{noise_intensity} = \frac{\max(n) - n}{\max(n) - \min(n)} \quad (3)$$

where *noise_intensity* denotes the intensity of noise, n is the value of the noise, $\max(n)$ represents the maximum value of the noise, $\min(n)$ represents the minimum value of the noise.

- (5) Use Equation (4) to calculate the dynamic noise and use Equation (5) to add noise to the image and to ensure that the pixel values remain within the valid range (0, 255), otherwise the resultant pixel values are clipped.

$$\text{dynamic_noise} = n \cdot \text{noise_intensity} \quad (4)$$

where *dynamic_noise* denotes the added noise.

$$\text{noise_image_array} = \text{image_array} + \text{dynamic_noise} \quad (5)$$

where *noise_image_array* denotes the image after adding noise, *image_array* denotes the image to which the noise is added.



Figure 6. Raw images before and after adding noise: (a–c) are the raw images; (d–f) are the noise-added images.

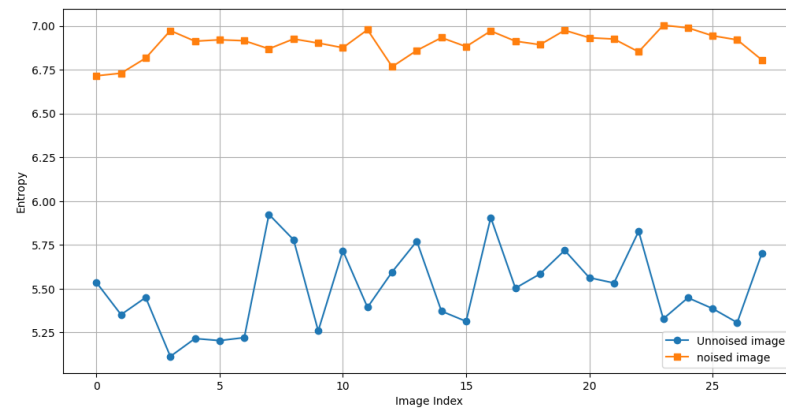


Figure 7. Comparison of images before and after adding noise.

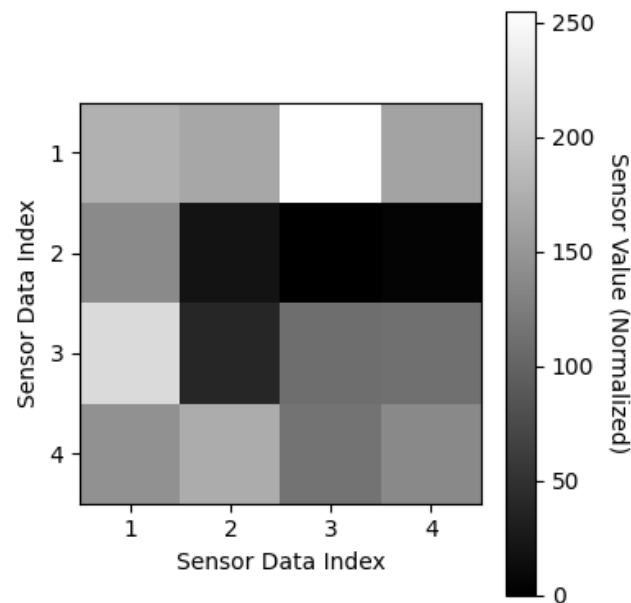


Figure 8. The noise in 2D.

3.3. DCGAN Model

A GAN utilizes a zero-sum game to learn the distribution from training sets. Figure 9 demonstrates the structure of a GAN that consists of a generator (G) and a discriminator (D). The G uses random inputs to generate fake samples with a distribution consistent with the real sample. The D can try to distinguish the fake samples from the outputs. The complete process can be described in Equation (6). The G can continuously generate until the D cannot distinguish whether the outputs are real or fake.

We used a DCGAN as our neural network model, which consisted of a GAN and a convolutional neural network (CNN). When compared with the traditional GAN, a DCGAN employs CNNs, which are inherently more efficient in processing image data. Due to the powerful feature extraction capability of CNNs, the learning efficiency of the GAN is greatly improved.

Figure 6 shows the generator and discriminator configuration. The generator was designed to upscale from the latent space dimension to produce an image. It employed a series of convolutional transpose layers (ConvTranspose2d) for upsampling and a batch normalization layer (BatchNorm2d) for feature normalization, which helped to address the vanishing gradient problem, expediting convergence and improving the fidelity of generated images. ReLU activation functions were used to take into account the computational efficiency while

introducing nonlinearity and selectivity, and a final layer used a Tanh activation function to limit the output to a range between -1 and 1 . The generator configuration is presented in detail in Table 3.

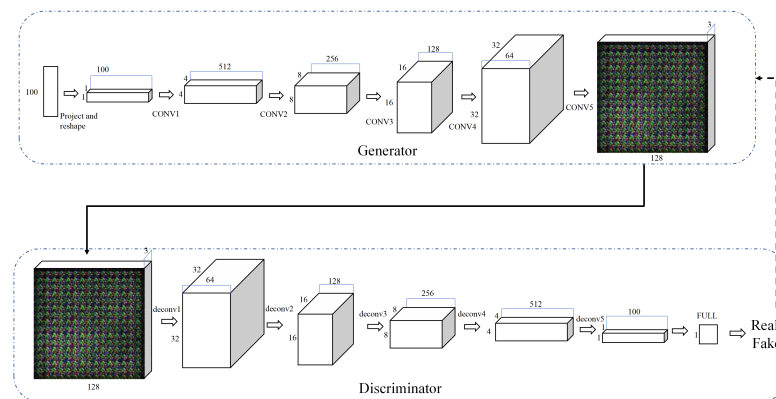


Figure 9. DCGAN network model.

Table 3. Layer configuration of the generator.

Layer	Activation Function	Kernel Num	Kernel Size	Stride	Padding
ConvTranspose2d	ReLU	512	4	1	0
BatchNorm2d	-	512	-	-	-
ConvTranspose2d	ReLU	256	4	2	1
BatchNorm2d	-	256	-	-	-
ConvTranspose2d	ReLU	128	4	2	1
BatchNorm2d	-	128	-	-	-
ConvTranspose2d	ReLU	64	4	2	1
BatchNorm2d	-	64	-	-	-
ConvTranspose2d	Tanh	3	4	2	0

Conversely, the discriminator downsampled the input image through convolutional layers, utilizing LeakyReLU activation functions to allow for a small, non-zero gradient when the unit was not active, thereby mitigating the vanishing gradient problem. It culminated in a sigmoid output, representing the probability of the input being a real image. This structure enabled the generator to synthesize high-fidelity images, while the discriminator effectively discerned real from generated images, optimizing the adversarial training process. The discriminator configuration is presented in detail in Table 4.

Table 4. Layer configuration of the discriminator.

Layer	Activation Function	Kernel Num	Kernel Size	Stride	Padding
Conv2d	LeakyReLU	64	4	2	0
Conv2d	LeakyReLU	128	4	2	1
BatchNorm2d	-	128	-	-	-
Conv2d	LeakyReLU	256	4	2	1
BatchNorm2d	-	256	-	-	-
Conv2d	LeakyReLU	512	4	2	1
BatchNorm2d	-	512	-	-	-
Conv2d	Sigmoid	1	4	1	0

The architecture of the DCGAN and its training methods enabled a more stable training process, reducing model collapse, and typically producing higher-quality images compared to those generated by traditional GANs.

$$\min_G \max_D V(G, D) = E_{x \sim p_{data}(x)} \log D(x) + E_{z \sim p_{data}(z)} \log(1 - D(G(z))) \quad (6)$$

where $x \sim p_{data(x)}$ represents the sample x collected from the real data distribution $p_{data(x)}$, while $z \sim p_{data(z)}$ denotes the random noise vector z sampled from a noise distribution $p_{data(z)}$. The generator G is trained to produce fake data with z that mimics x as closely as possible, whereas the discriminator D is trained to distinguish between real data and fake data generated by G . The term $\min_G \max_D V(G, D)$ refers to the adversarial value function in the GAN. $E_{x \sim p_{data(x)}} \log D(x)$ represents the expected value of the $\log$ probability that the D correctly classifies a real data sample x as real.

Conversely, $E_{z \sim p_{data(z)}} \log(1 - D(G(z)))$ represents the expected value of the $\log$ probability that the D correctly identifies the data $G(z)$ generated by the G from a noise input z as fake.

3.4. Entropy-Driven Stopping Method

The proposed entropy-driven stopping method was designed to overcome the overfitting of the GAN-based PRNG. To identify the optimal epoch for stopping the network training process, the average image entropy and the variance of the image entropy of the training were monitored.

Theoretically, a random sequence needs to satisfy at least the following properties:

- (1) Uncertainty: each code is unpredictable, and the unknown code cannot be calculated by using other known codes.
- (2) Independence ensures that no relationship exists between the codes in the sequence and prevents the possibility of deducing the future value from previous ones.
- (3) Bias-free: There is no bias in the sequence; if the random numbers tend toward some certain values, the sequence of random numbers is biased.

The purpose of using a DCGAN is not only to imitate the distribution of the training set but also, more importantly, to learn the randomness and generate images that possess sufficient randomness. According to these specific properties, if an image has a wide variety of pixel values that appear with nearly equal probability, we can infer with high confidence that the pixels are randomly distributed. Therefore, we can use image entropy as an indicator to assess the randomness of the image.

Analogous to the early-stopping method, the proposed entropy-driven stopping method also allows for a more flexible determination of the stopping epoch. Our method introduces a patience parameter, denoted as p to identify the optimal stopping point in the learning process. Training continues until the rolling image entropy of the training data reaches the specified threshold parameters (the average image entropy e and its variance v for a minimum of e_{min} epochs). This relaxed approach of determining the stopping point, referred to as L_{stop} , facilitates a more accurate selection of when to halt the training.

Therefore, the suggested approach aims to rectify the uncontrolled generation of high-entropy images by the DCGAN. To determine the optimal moment to stop the training process of the DCGAN, we monitor e and v . Figure 10 illustrates a diagram of the stopping criterion. During training, the rolling image entropy with window size w is calculated over 4 epochs for a step during the training process. At epoch L_{min} , if the image entropy exceeds the threshold e and the variance of the image entropy is below threshold e , the training process persists for an additional $4w$ epochs to guarantee that the DCGAN does not produce a superior model. Conversely, training halts at epoch L_{stop} if the assessed rolling image entropy exceeds e and the variance of the image entropy remains below e for $4w$ epochs. At that point, the model that best generates images with high entropy is selected, denoted by L_{best} . Additionally, the algorithm can be mainly described with the following steps:

- (1) Use image entropy to assess the randomness of generated images. Moreover, add variance entropy to ensure the model can generate image stably.

- (2) If the average image entropy reaches the target value e , calculate the variance in entropy of the generated images to evaluate whether the model can consistently and stably produce images with a variance v , ensuring they maintain the target entropy.
- (3) If the model can generate images with the e and v , stop the training process. If the model can generate images with the target entropy and a low entropy variance, it indicates that the model can consistently provide inputs with high security for the proposed PRNG, at which point we stop the training process.
- (4) The training with the monitoring algorithm is detailed in Algorithm 2.

Algorithm 2: Entropy-driven stopping method

Input: training_sets t ; inputs z ; target_entropy e ; target_variance_of_entropy v ; rolling_window w ; patience p

Output: DCGAN model M ; training_epoch L_{best} ;

```

1 Initialize the parameters ;
2 Initialize generator  $G$  and discriminator  $D$ ;
3  $L = 0$ ;
4 for  $i = 1$  to  $max\_epoch$  step  $w$  do
5   fake  $\leftarrow G(z)$ ;
6   real  $\leftarrow D(x)$ ;
7   //Train the DCGAN model;
8   train(real, fake);
9   Update the parameters of  $G$  and  $D$  by gradient descent;
10   $L = L + 1$ ;
11  //Calculate the average of image entropy and variance of image entropy;
12   $en \leftarrow \text{Entropy}(\text{fake})$ ;
13   $ven \leftarrow \text{Variance\_of\_entropy}(\text{fake})$ ;
14  if  $en \geq e$  and  $ven \leq v$  then
15    for  $j = 1$  to  $p$  do
16      train(real, fake);
17       $L = L + 1$ ;
18       $en \leftarrow \text{Entropy}(\text{fake})$ ;
19       $ven \leftarrow \text{Variance\_of\_entropy}(\text{fake})$ ;
20      if  $en \leq e$  and  $ven \geq v$  then
21        break;
22        //Pass unsuccessfully and go to next window;
23      else
24        continue;
25        //Pass successfully and go to next epoch;
26         $L_{list}.append(L)$ ;
27        //Collect the information of  $L$  under the patience;
28      end
29    end
30     $L_{best} = \text{Best}(L_{list})$ ;
31    //Search the best  $L$ ;
32  else
33    Update the parameters of  $G$  and  $D$  by gradient descent;
34  end
35 end
36 return  $M, L_{best}$ ;
```

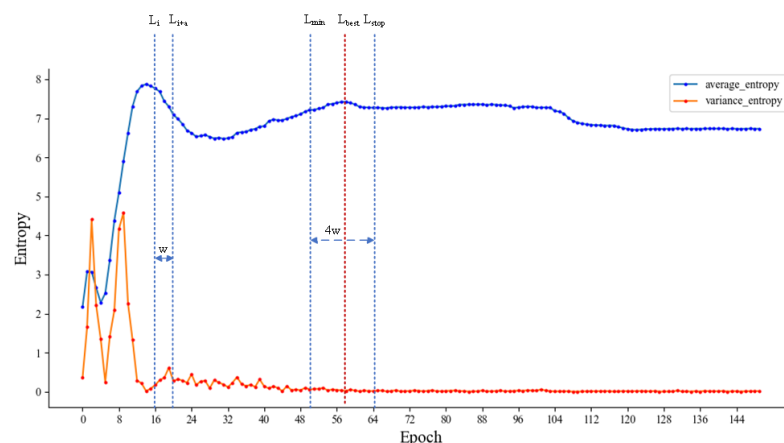


Figure 10. The visual representation of the entropy-driven stopping method.

3.5. Postprocessing

Chaos mapping is a mathematical function that describes random behavior. It is very sensitive to changes in inputs. Even small changes cause the mapping to be very different. A logistic map is one of the most deeply studied chaotic maps in chaotic cryptography, and its mathematical formula is described in Equation (7):

$$x_{n+1} = mx_n(1 - x_n) \quad (7)$$

where n is the iteration time, for any t , $x_n \in [0, 1]$, m is an adjustable parameter to ensure that the mapped x_n is always located within $[0, 1]$, then $m \in [0, 4]$. x_n is the ratio of the population to the maximum possible population size at time n .

By varying the parameter m , the Equation (7) exhibits different dynamical limit behaviors as n approaches infinity. These behaviors include fixed points, periodic orbits, or chaotic states. As shown in [38], when $3.5 < m \leq 4$, the Logistic map becomes chaotic, as illustrated in Figure 11, n is the iteration times of the logistic map, x_n is the output of the logistic map.

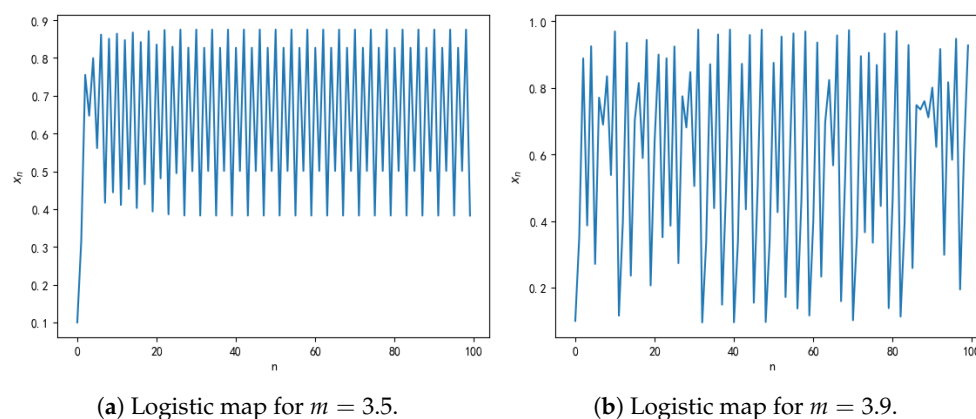


Figure 11. Logistic map visualization.

In this article, we set the initial value of m as 3.9 to control the behavior of the logistic map, and Equation (8) was used to control the parameters m . During the experimental process, m was updated at each iteration to ensure the chaos of the logistic map. The parameter c was used to help in maintaining the chaos of the logistic map.

$$m_i = m_{i-1} + 0.001x + c \quad (8)$$

Some researchers proposed to use Chaotic Traveling Waves in a Coupled Map Lattice (CCML) to further process the images to generate random bits [24,25]. Therefore, according to these existing studies, we used the image postprocessing method proposed in [25] to process the generated high-entropy images to generate random numbers, with an appropriate size of the lattice of 5 or 6. The mathematical formula is depicted in Equation (9).

$$x_{n+1}^i = af(x_n^i) + \frac{1-a}{2}(f(x_n^{i+1}) + x_n^{i-1})) \quad (9)$$

where a represents the coupling coefficient, which ranges in $[0, 1]$, the function $f(x_n^i)$ denotes a chaotic map, where i is the index of the chaotic maps, each of which can be referred to as a lattice site in the CCML. The postprocessing method can be mainly described with the following steps:

- (1) Initialize the parameters of the logistic map and the CCML to process the generated images into pseudo-random numbers: (i) set the coupling coefficient of the CCML to 0.5, (ii) set a 32-bit mask for the later operation, (iii) set an empty list to store the results.
- (2) Iterate through each image to generate a sequence and normalize the data values to a specific range, ensuring consistency in size.
- (3) Use the CCML and the logistic map to process these data to generate new data for each lattice site of CCML, and extract some significant bits from the generated data.
- (4) Use XOR operations to generate the output value and re-encode the output in binary to obtain a text as the pseudo-random numbers.

4. Experiment and Security Analysis

The datasets were 500 randomly chosen images and other sensor data from KIITI [37]. Firstly, we analyze the security of the proposed PRNG in Section 4.1, then we assess the proposed early stopping method and recording the stopping point in Section 4.2. Furthermore, we use statistical test suites (NIST [18], ENT [19]) to test the random numbers generated by the proposed methods in Section 4.3. Finally, to assess the performance in encryption applications of the proposed PRNG, we calculate the NPCR, UACI and adjacent pixel correlation of the images encrypted by the proposed pseudo-random numbers in Section 4.4.

4.1. Confidentiality Analysis

The public chaos algorithms of the traditional PRNGs have some potential security risks: (i) they can be attacked by inverse methods and (ii) fixed inputs can be predicted by attacks. The GAN-based PRNGs in previous research can overcome these shortcomings. However, these methods still have some security risks: (i) the training sets usually rely on public chaos; (ii) the training of the GAN is usually uncontrolled, and excessive training epochs can result in overfitting of the network.

In response to such security challenges, an effective PRNG is proposed in this paper. We use the data collected by vehicle sensors as training sets, and we propose a pre-processing method to process these data to generate images with strong randomness that can serve as excellent training sets for the DCGAN. Therefore, the proposed PRNG system is independent of the public chaos algorithm. Furthermore, we design an entropy-driven stopping method to control the training, which help to prevent overfitting and conserve computational resources.

In addition, we use a fog service to transmit data. As a result, when some vehicles are attacked and unable to collect any data to send to the IoV, the PRNG system on the IoV remains operational despite potential leaks of pseudo-random numbers to attackers. Other vehicles can continue to gather data and transmit them to the IoV, enabling the generation of pseudo-random numbers. Furthermore, the hacked vehicles can use the pseudo-random numbers generated by fog service to update their PRNG systems, ensuring the encryption system keeps running normally to wait for the PRNG system to be fixed.

In conclusion, by utilizing data captured by vehicle sensors as training sets and leveraging the network architecture of fog services, we significantly enhance the security and reliability of the PRNG system. This approach ensures that even if certain nodes are attacked, the overall encryption system within the IoV remains operational. Furthermore, it not only strengthens the defense capabilities of the PRNG system, but also enhances its stability and robustness for continuous operation.

4.2. Early Stopping

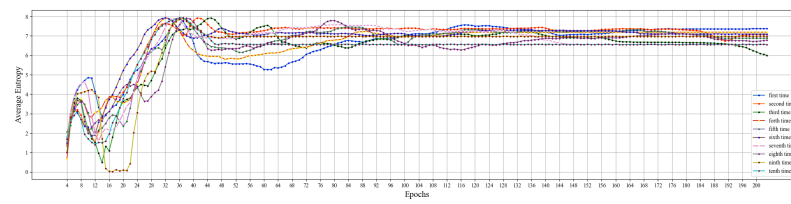
4.2.1. Entropy Analysis

To explore the output patterns of the DCGAN during the training process, we used image entropy and its variance of image entropy as indicators to evaluate the model's effectiveness. In our experiments, the DCGAN was run 10 times, each for 200 epochs. At each evaluation point, we built 20 models, and each model generated 100 images. For each epoch, we calculated the average entropy and the entropy variance of the generated images, providing a comprehensive assessment of the model's output quality over time.

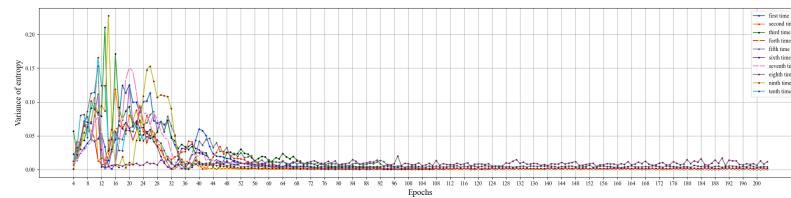
The average image entropy over 200 epochs is shown in Figure 12a, while the variance of the entropy is presented in Figure 12b. We observed that their relationship was not strictly proportional. The average entropy values first increased, then decreased and increased again with training, and when the training epoch reached a certain number of epochs, the average entropy value gradually climbed and stabilized. Meanwhile, the variance of the entropy value initially exhibited strong fluctuations, and when the training epoch reached a certain number of epochs, the variance of the entropy began to decrease and stabilized.

Furthermore, as training continued, we observed a significant drop in average image entropy values, suggesting potential overfitting. Therefore, to explore whether overfitting occurred during the training process, we extended the training to 2000 epochs. The results are shown in Figure 12c,d, where we observe a sharp decline in average image entropy between 200 and 300 epochs, indicating a degree of overfitting in prolonged training. Although combining average image entropy with the variance of the entropy allowed us to identify the optimal training point (where the average entropy is high and the variance of entropy is low), it remained challenging to consistently train a model at the same epoch that could stably generate high-entropy images.

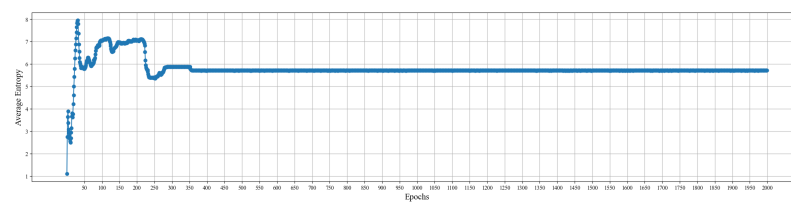
From the above analysis, we observe that extended training led to overfitting, as indicated by the significant drop in average image entropy after a certain number of epochs. This overfitting not only diminished the model's ability to generate high-entropy images but also resulted in unnecessary computational costs due to excessive training. To mitigate this, an early-stopping algorithm could help by halting the training process when the model reaches an optimal point, thereby avoiding overfitting and conserving computational resources.



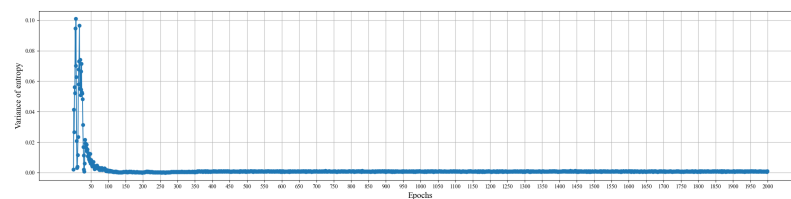
(a) The average entropy over 200 epochs.



(b) The variance of the entropy over 200 epochs.



(c) The average entropy over 2000 epochs.



(d) The variance of the entropy over 2000 epochs.

Figure 12. The statistics of image entropy.

4.2.2. Evaluation of the Entropy-Driven Stopping Method

Figure 13 shows the training sets of the proposed PRNG. The GPS module was used to collect the longitude, latitude, and height, while the IMU module collected the velocity and acceleration. Additionally, the parameter controller set the accuracy parameters for both IMU and GPS data collection. Current early-stopping criteria are not tailored specifically for GAN-based PRNGs. These methods determine the knee point based on training losses rather than evaluating the randomness of the outputs. Therefore, to evaluate the effectiveness of the proposed entropy-driven stopping method described in Section 3.4, we compared it with the traditional maximum number of epochs stopping method, which limits network training by a predetermined number of epochs. As discussed in Section 4.2.1, an effective model should produce high-entropy images with minimal entropy variance while also reducing training epochs to conserve computational resources, so our assessment focused on three critical aspects: average entropy (representing the randomness of outputs), variance of the entropy (indicating the stability of outputs), and the number of stopping epochs (reflecting resource costs).

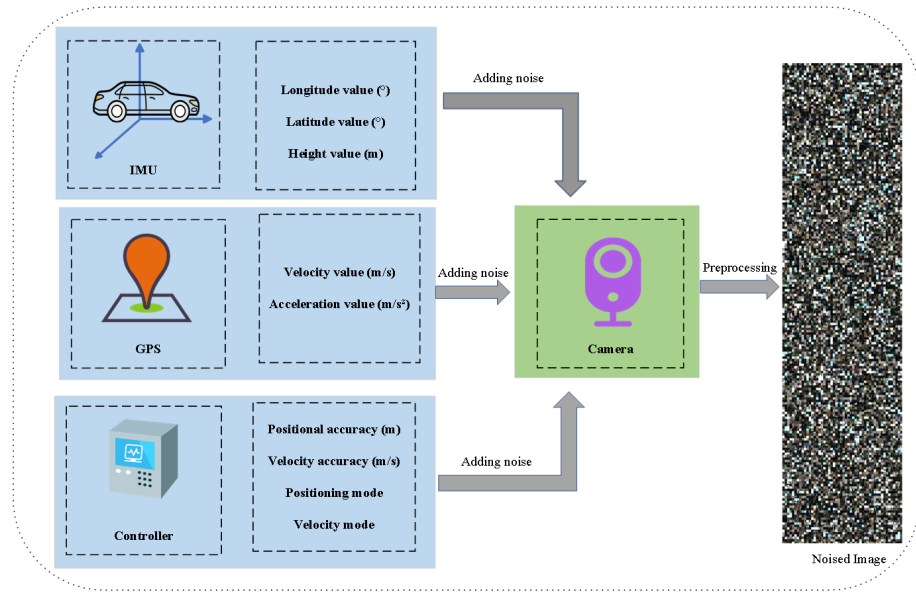


Figure 13. The datasets collected from multi-sensors in vehicle.

Consequently, we designed an effectiveness score to evaluate the effectiveness of the entropy-driven stopping method, as described in Equations (10)~(12). The score R comparing the maximum early stopping method with our entropy-driven stopping method is visualized in Figure 14.

$$R = k_T T + k_m \frac{E_m}{E_{best}} \quad (10)$$

$$E_m = E_0 / V_0 \quad (11)$$

$$E_{best} = E_1 / V_1 \quad (12)$$

where R represents the effectiveness score, while T denotes the stopping epoch. E_0 is the image entropy and V_0 is the variance of the image entropy at the point of stopping. E_1 refers to the image entropy, and V_1 is the variance of the image entropy when monitoring the highest E_{best} during the training process. k_T and k_m are weights used in the equation, and we set $k_T = 0.1$ and $k_m = 1$ in this article.

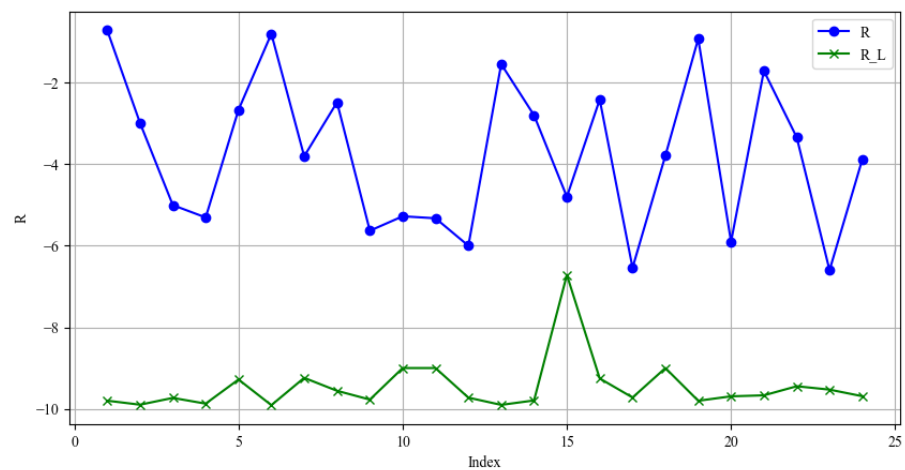


Figure 14. Comparison of R before and after early stopping.

The experiment results showing the comparison of R between the maximum-stopping method and the proposed early-stopping method are illustrated in Figure 14. The green line represents the R values from the maximum-epoch-stopping method, while the blue

line corresponds to those from the proposed early-stopping method. It is evident that values from our early-stopping method significantly exceeded those of the maximum-epoch method, indicating that our approach could generate high-entropy images reliably while minimizing the number of training epochs. Furthermore, this suggested that the DCGAN could generate similar or even superior quality data in fewer epochs, thereby greatly reducing computing resource requirements. Figure 15 shows the results of the DCGAN outputs, using the proposed entropy-driven stopping method. Additionally, to evaluate the generalizability of the proposed entropy-driven stopping method, we applied it to the datasets from [39,40]. The results, presented in Figure 16a,b, demonstrated that the method effectively improved performance across different datasets. These findings validate the adaptability of our method, confirming its effectiveness in enhancing DCGAN performance under widely varying conditions.

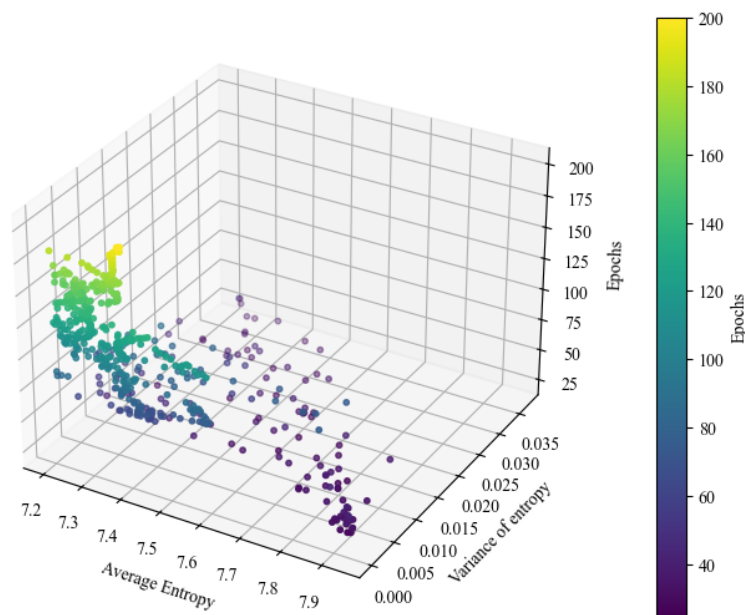
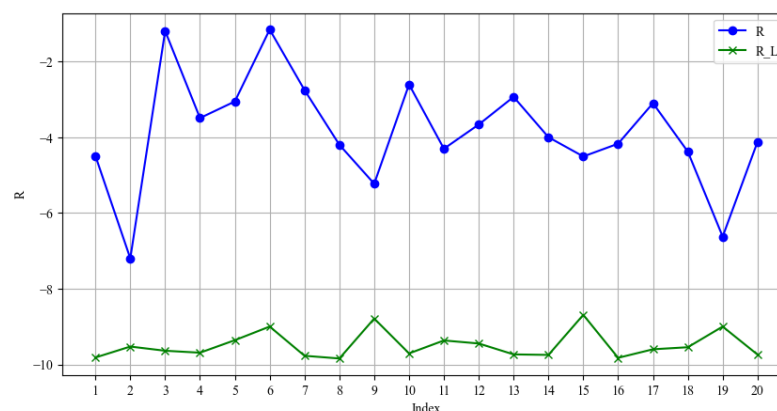


Figure 15. The optimal training epochs.



(a)

Figure 16. Cont.

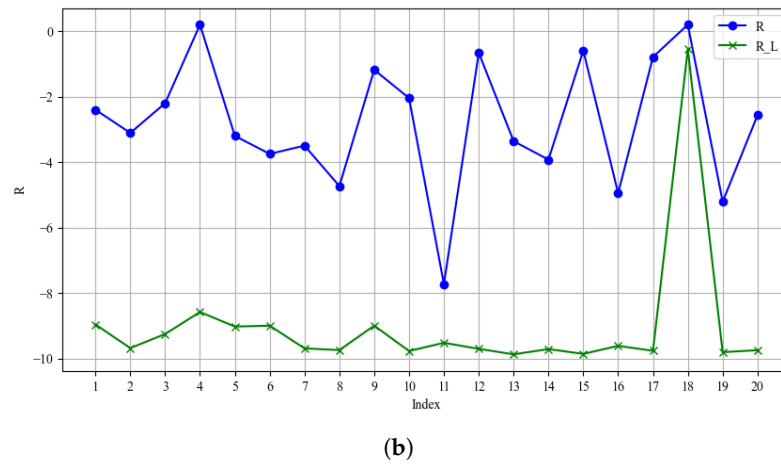


Figure 16. Comparison of R before and after early stopping on the other datasets. (a) Comparison of R before and after early stopping on the BDD100k dataset; (b) Comparison of R before and after early stopping on the Tsinghua-Tencent100K dataset.

4.2.3. Analyzing the Parameters of the Entropy-Driven Stopping Method

After evaluating the effectiveness of the entropy-driven stopping method, we further investigated the preset parameters' influence on the proposed entropy-driven stopping method.

The influence of ϵ was discussed in Section 4.2.2, where it was identified as a necessary requirement for randomness. In this analysis, we focus on the patience p and the variance of the average entropy v . A lower variance and higher patience suggest that the generated model exhibits greater stability, while such stability may also imply that more training epochs are required to accurately identify the optimal early stopping point. The influences are illustrated in the Figure 17.



Figure 17. Optimal L_{best} with different p and v parameters.

Figure 17 illustrates that as p increased and v decreased, the epoch of the optimal early stopping point (L_{best}) also increased. Moreover, the noticeable patterns observed in the parameter space strongly suggested that it was feasible to develop a method for identifying the optimal parameters. In particular, when $p \geq 10(2.5w)$ and $v \leq 0.01$, L_{best} showed a significant increase. Therefore, to ensure as strict thresholds as possible while minimizing

L_{best} , we used the boundaries where there was a noticeable color change in Figure 17 as the optimal parameters ($p_{best} = 10$, $v_{best} = 0.01$).

Furthermore, to more accurately locate the optimal parameters, Figure 18a,b illustrate how the average optimal epochs changed as the variance and patience varied and show the sensitivity of the optimal epochs. In Figure 18a, when v decreases from 0.015 to 0.01, the optimal epoch shifts by 36. Since the value of v at that point was relatively large, we chose $v_{best} = 0.015$ as the optimal parameter. Similarly, in Figure 18b, while the optimal epoch for p varied significantly between 10 to 12 and 14 to 16, we chose $p_{best} = 14$ as the optimal parameter to achieve a relatively larger p .

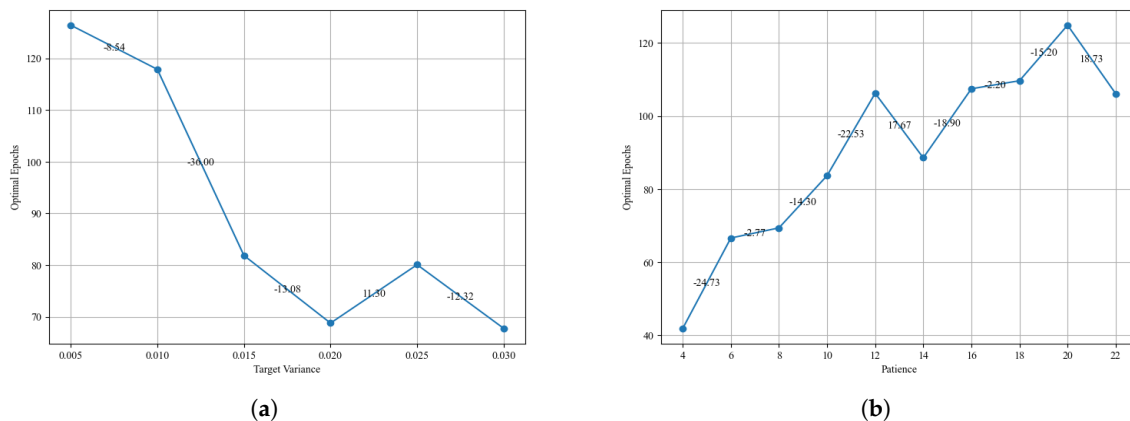


Figure 18. The sensitivity of the optimal epoch. (a) The sensitivity of the optimal epoch when the target variance changes; (b) The sensitivity of the optimal epoch when the patience changes.

4.3. Tests on Statistical Suites

A high entropy value with the generated image does not completely imply that it can serve as a reliable source of entropy. Therefore, to further evaluate the randomness of the proposed PRNG, we used the NIST 800-22 and ENT test suites to test the pseudo-random sequence extracted from the generated pseudo-random numbers.

We stitched together high-entropy images of size 128×128 generated by the proposed DCGAN to form a large high-entropy image of size 1280×1280 . Then, we applied a chaotic postprocessing algorithm to process the stitched images to generate a pseudo-random sequence of 38 MB in size.

Table 5 lists 15 results of the randomness tests in the NIST 800-22 test suite; the p -value is an indicator to test the uniformity of the distribution of the sequence, where if the p -value is greater than 0.01, it means that the test has been passed. The generated pseudo-random numbers passed all tests, and each test pass rate exceeded 95%. This demonstrates that the generated pseudo-random numbers exhibited strong statistical properties. Furthermore, since these pseudo-random numbers were derived from a set of images generated by the same model, it implied that there was virtually no correlation among the images produced by that model.

Table 6 presents the results of the randomness tests in the ENT test suite. The pseudo-random sequences passed all tests in ENT, indicating a high level of statistical randomness. In particular, the sequences were already in their most compressed form and could not be further compressed. Additionally, the entropy value was very close to eight, suggesting that each possible byte value (ranging from 0 to 255) occurred with nearly equal probability. The correlation coefficient was nearly zero, indicating no linear relationship between consecutive data points in the sequence.

Therefore, users just need to use the pre-trained model trained by the PRNG developers to generate pseudo-random numbers, and the proposed PRNG can strongly protect the security of information on the IoV.

Table 5. The NIST test results.

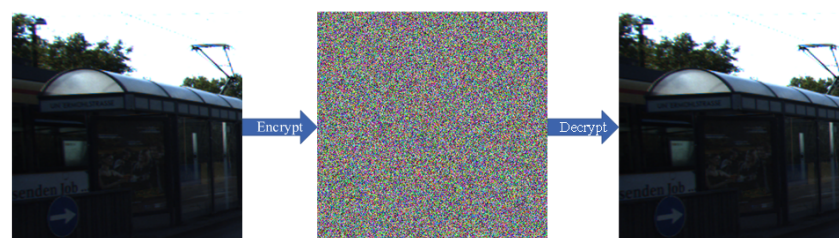
Test Name	<i>p</i> -Value	Pass-Rate	Result
Frequency	0.102526	98/100	pass
BlockFrequency	0.455937	100/100	pass
CumulativeSums	0.148808	98/100	pass
Runs	0.419021	97/100	pass
LongestRun	0.739918	99/100	pass
Rank	0.249284	99/100	pass
FFT	0.911413	98/100	pass
NonOverlappingTemplate	0.527246	99/100	pass
OverlappingTemplate	0.383827	99/100	pass
Universal	0.534146	98/100	pass
RandomExcursions	0.367754	99/100	pass
ApproximateEntropy	0.699313	98/100	pass
RandomExcursionsVariant	0.401408	99/100	pass
LinearComplexity	0.419021	100/100	pass
Serial	0.709530	99/100	pass

Table 6. The ENT test results.

Test Name	Test Output	Result
Entropy	7.999995	pass
Optimum compression	0%	pass
Chi-square	245.48 (65.44%)	pass
Arithmetic mean	0.148808	pass
Monte Carlo value for π	3.142666626 (0.03%)	pass
Serial correlation coefficient	−0.000097	pass

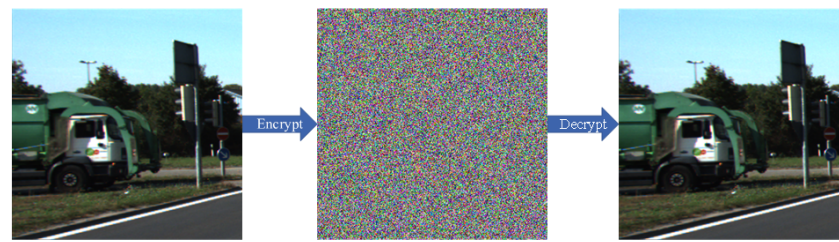
4.4. Application Evaluation

Figure 19 illustrates the images before encryption, after encryption, and after decryption. The encrypted images show that when the image has been encrypted, there is almost no obvious pattern between the pixels. The decrypted images show that the original image can be recreated with almost no loss in quality.

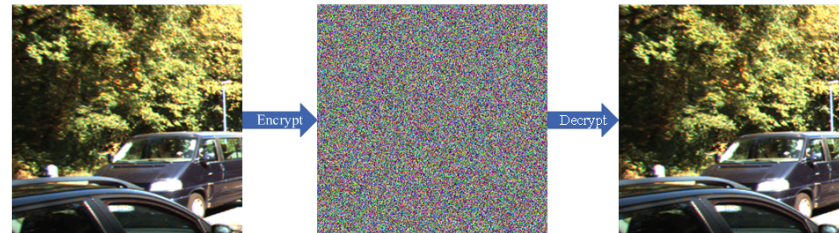


(a) Encryption and decryption processes of the first image.

Figure 19. Cont.



(b) Encryption and decryption processes of the second image.



(c) Encryption and decryption processes of the third image.

Figure 19. The encryption and decryption processes of the images.

4.4.1. Resistance Differential Attack Analysis

To further ensure the security of the PRNG in applications, we used the generated pseudo-random numbers to encrypt two images with only one-pixel difference and calculated their UACI and NPCR. NPCR and UACI can evaluate the strength of image encryption algorithms against differential attacks. UACI can be calculated as the average intensity changes between the two images, while NPCR evaluates the proportion of pixels that differ between the two images. The ideal value of NPCR is 99.6094%, and the ideal value of UACI is 33.4635%; this implies that the closer the test results are to the ideal values, the higher the security of the proposed PRNGs. Furthermore, Table 7 demonstrates that the UACI and NPCR of images encrypted using our EPRNG achieved the 0.05 significance level, confirming strong resistance to attacks. The simulation steps were as follows.

Firstly, we randomly selected 1 MB of data from the generated random numbers to use as the encryption key. Secondly, we randomly downloaded a 128×128 PNG image from the internet; the histogram of that image is shown in Figure 20a. Then, we altered one pixel in the image to evaluate the algorithm's resistance to differential attacks, and we used the XOR operation on this pair of images with the encryption key generated by the pseudo-random numbers; the histogram of the ciphered image is shown in Figure 20b. Finally, we calculated the NPCR and UACI for the image pair.

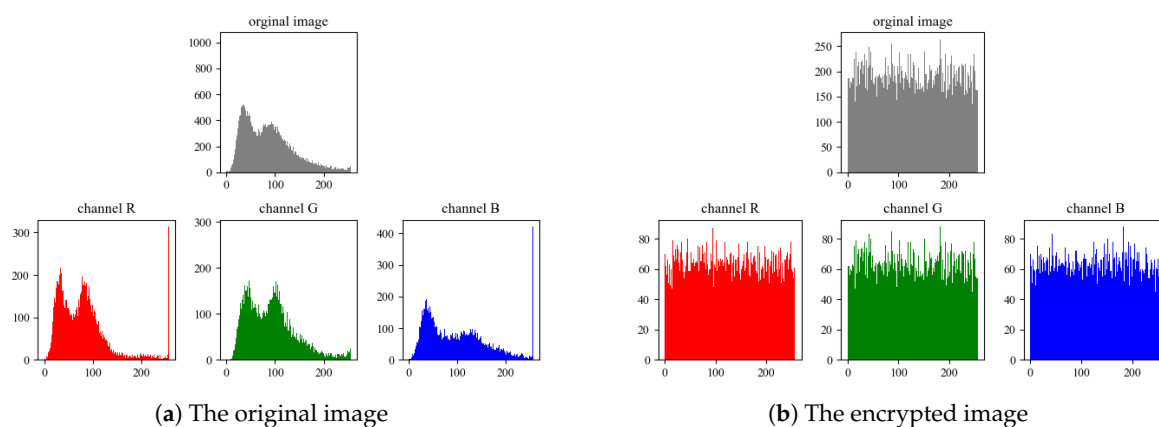


Figure 20. The histogram of the image pair.

Table 7. UACI and NPCR test results.

File	NPCR	UACI
File1	99.6826%	33.2007%
File2	99.5605%	33.4853%
File3	99.6399%	33.8457%
File4	99.6227%	33.6958%
Optimal value	$NPCR_{0.05} \geq 99.5292\%$	$33.1012\% \leq UACI_{0.05} \leq 33.8259\%$
	$NPCR_{0.01} \geq 99.3826\%$	$32.9874\% \leq UACI_{0.01} \leq 33.9397\%$
	$NPCR_{0.001} \geq 99.3082\%$	$32.8552\% \leq UACI_{0.001} \leq 34.0718\%$

4.4.2. Adjacent Pixel Correlation Analysis

One of the criteria for assessing the quality of image encryption algorithms is the adjacent pixel correlation analysis. This method computes the correlation coefficient between neighboring pixels in different directions, such as horizontal, vertical, or diagonal.

The correlation coefficient is a measure of the linear relationship between two variables, ranging from -1 to 1 . A high correlation coefficient implies that the pixels have similar values, while a low correlation coefficient implies that the pixels have different values. An ideal image encryption algorithm can generate an image that has a low correlation coefficient for any pair of adjacent pixels, regardless of the direction. The correlation coefficient can be calculated by Equations (13)~(16).

$$E(x) = \frac{1}{N} \sum_{i=1}^N x_i \quad (13)$$

where x is the dataset of the pixel values for an image and N represents the total number of pixels in the image. $E(x)$ can be calculated as the average of all x_i 's

$$D(x) = \frac{1}{N} \sum_{i=1}^N (x_i - E(x))^2 \quad (14)$$

where $D(X)$ provides a measure of how much the data points deviate from $E(x)$, a higher $D(x)$ indicates greater spread, suggesting that the data points are more dispersed.

$$\text{Cov}(x, y) = \frac{1}{N} \sum_{i=1}^N (x_i - E(x))(y_i - E(y)) \quad (15)$$

where y is the datasets of pixel values of x adjacent pixels, and $\text{Cov}(x, y)$ is a measure of the relationship between x and y .

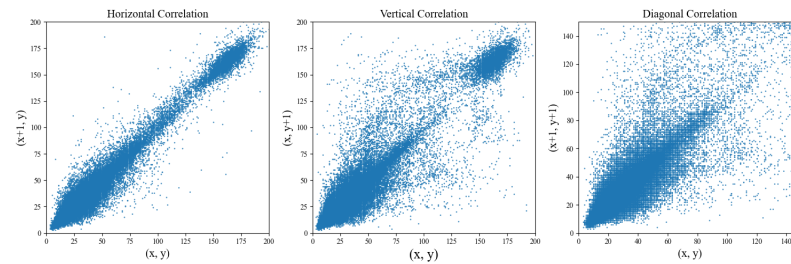
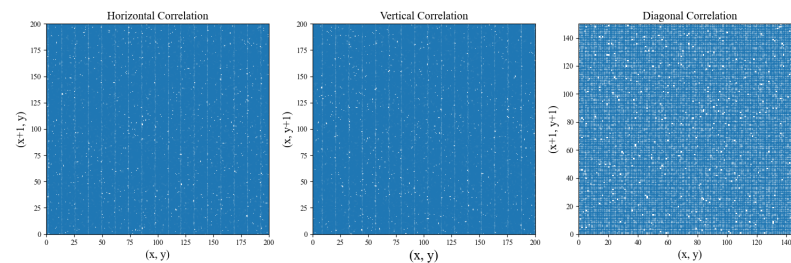
$$r_{xy} = \frac{\text{cov}(x, y)}{\sqrt{D(x)D(y)}} \quad (16)$$

where r_{xy} measures the strength and direction of the linear relationship between x and y . It yields a value between -1 and 1 . The closer r_{xy} is to zero, the weaker the correlation between x and y .

Figure 21 and Table 8 demonstrate that the correlation value between adjacent pixels in the original image was very close to one, indicating a strong correlation between nearby pixels. In contrast, the correlation value in the encrypted image was nearly zero, suggesting that there was almost no correlation between adjacent pixels. In particular, Table 8 shows that the pixel correlation in the encryption using our proposed method was even lower than that of other articles [41–43], indicating a higher level of security.

Table 8. Correlation coefficients of the cipher-images

Method	Horizontal	Vertical	Diagonal
[41]	0.0093	0.0159	0.0097
[42]	−0.002097	0.016187	0.017805
[43]	0.0035	0.0179	−0.0128
Proposed method	0.0056	−0.0031	0.0013
Original image	0.9922	0.9889	0.9841

**(a)** The adjacent pixel correlation of the original image.**(b)** The adjacent pixel correlation of the encrypted image.**Figure 21.** The adjacent pixel correlation of the image pair.

5. Conclusions

In this paper, we proposed a novel PRNG based on a DCGAN to address the challenge of security on the IoV with the consideration of environmental interference and training dynamics. We designed a preprocessing method integrating image stitching and noise adding to combine images from extracted high-entropy pixel blocks and add data from other vehicle sensors as image noises, which could defend against environmental interference. We presented an entropy-driven stopping method that evaluated the image entropy during the training process of the DCGAN, while enabling the determination of the optimal number of epochs. The randomness of our approach was validated through extensive numerical experiments using statistical test suites, demonstrating that our DCGAN-based RNG produced high-quality random numbers suitable for IoV applications. Moreover, we successfully applied the generated random numbers to image encryption, achieving performance metrics that closely approximated ideal values. This demonstrated the reliability and robustness of our RNG in practical security applications on the IoV. In short, our research provides a significant contribution to the field of cryptography-based protocols for the IoV, offering a practical and efficient solution for PRNGs based on GAN. In future work, we mean to develop data-sharing schemes for increasingly complex IoV systems. To enhance both the efficiency and security of data transmission, another extension of our work is to integrate edge computing into the system and introduce blockchain technology to safeguard data security. These improvements will enable the proposed PRNG to be better suited for more complex systems with higher security demands.

Author Contributions: C.F.: conceptualization of this study, validation, methodology, software, data curation, writing—original draft. X.Z.: conceptualization of this study, validation, methodology, software, data curation, writing—review and editing, funding acquisition. D.W.: methodology, software, data curation. H.H.: methodology, data curation. R.H.: methodology, data curation. Z.W.: methodology, data curation. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the “National Natural Science Foundation of China”, grant number 61802252.

Institutional Review Board Statement: Not applicable. The study did not involve humans or animals. We used data from a third party.

Informed Consent Statement: Not applicable.

Data Availability Statement: Restrictions apply to the availability of these datasets. The first dataset was obtained from A. Geiger et al. [37] and is available at <https://doi.org/10.1109/CVPR.2012.6248074> (accessed on 1 May 2024) with permission from A. Geiger et al. The second dataset was provided by Yu et al. [39] and can be accessed at <https://bair.berkeley.edu/blog/2018/05/30/bdd/> (accessed on 1 December 2024). The third dataset, from Zhu et al. [40], is available at <https://cg.cs.tsinghua.edu.cn/traffic-sign/> (accessed on 1 December 2024).

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Duan, W.; Gu, J.; Wen, M.; Zhang, G.; Ji, Y.; Mumtaz, S. Emerging technologies for 5G-IoV networks: Applications, trends, and opportunities. *IEEE Netw.* **2020**, *34*, 283–289. [CrossRef]
2. Yang, H.; Feng, T.; Yan, Y. Implementing efficient attribute encryption in IoV under cloud environments. *Comput. Netw.* **2022**, *218*, 109363. [CrossRef]
3. Rabadi, N.M. Implicit certificates support in IEEE 1609 security services for Wireless Access in Vehicular Environment (WAVE). In Proceedings of the 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2010), San Francisco, CA, USA, 8–12 November 2010; pp. 531–537. [CrossRef]
4. Eckhoff, D.; Sofra, N.; German, R. A performance study of cooperative awareness in ETSI ITS G5 and IEEE WAVE. In Proceedings of the 10th Annual Conference on Wireless On-demand Network Systems and Services (WONS), Banff, AB, Canada, 18–20 March 2013; pp. 196–200.
5. Chen, Y.; Zhu, F.; Tian, Y.; Xu, S.; Han, L.; Zhou, Q.; Ling, N. LETRNG—A lightweight and efficient true random number generator for GNU/Linux systems. *Tsinghua Sci. Technol.* **2022**, *28*, 370–385. [CrossRef]
6. Sivaraman, R.; Rajagopalan, S.; Amirtharajan, R. FPGA-based generic RO TRNG architecture for image confusion. *Multimed. Tools Appl.* **2020**, *79*, 13841–13868. [CrossRef]
7. Zanolini, T.; Ranjan, A.; O’Shea, S.J.; Raghavan, N.; Thamankar, R.; Pey, K.L.; Puglisi, F.M. Guidelines for the design of random Telegraph Noise-based true random number generators. *IEEE Trans. Device Mater. Reliab.* **2024**, *24*, 184–193. [CrossRef]
8. Wan, Y.; Chen, K.; Huang, F.; Wang, P.; Leng, X.; Li, D.; Kang, J.; Qiu, Z.; Yao, Y. A flexible and stretchable bionic true random number generator. *Nano Res.* **2022**, *15*, 4448–4456. [CrossRef]
9. Yang, Y.G.; Zhao, Q.Q. Novel pseudo-random number generator based on quantum random walks. *Sci. Rep.* **2016**, *6*, 20362. [CrossRef]
10. Chen, T.; Ma, Y.; Lin, J.; Cao, Y.; Lv, N.; Jing, J. A lightweight full entropy TRNG with on-chip entropy assurance. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2021**, *40*, 2431–2444. [CrossRef]
11. Okada, K.; Endo, K.; Yasuoka, K.; Kurabayashi, S. Learned pseudo-random number generator: WGAN-GP for generating statistically robust random numbers. *PLoS ONE* **2023**, *18*, e0287025. [CrossRef] [PubMed]
12. Pasqualini, L.; Parton, M. Pseudo random number generation through reinforcement learning and recurrent neural networks. *Algorithms* **2020**, *13*, 307. [CrossRef]
13. Radford, A.; Metz, L.; Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv* **2015**, arXiv:1511.06434. [CrossRef]
14. Ciampi, L.; Gennaro, C.; Carrara, F.; Falchi, F.; Vairo, C.; Amato, G. Multi-camera vehicle counting using edge-AI. *Expert Syst. Appl.* **2022**, *207*, 117929. [CrossRef]

15. Kaneko, K. Chaotic traveling waves in a coupled map lattice. *Phys. D Nonlinear Phenom.* **1993**, *68*, 299–317. [CrossRef]
16. Li, J.; Madry, A.; Peebles, J.; Schmidt, L. Towards understanding the dynamics of generative adversarial networks. *arXiv* **2017**, arXiv:1706.09884. [CrossRef]
17. Makkadem, A. Estimation of the entropy and information of absolutely continuous random variables. *IEEE Trans. Inf. Theory* **1989**, *35*, 193–196. [CrossRef]
18. Rukhin, A.; Soto, J.; Nechvatal, J.; Smid, M.; Barker, E. A statistical test suite for random and pseudorandom number generators for cryptographic applications. In *NIST Special Publication 800-22*; NIST : Gaithersburg, MD, USA, 2001; Volume 800, p. 163. Available online: https://www.researchgate.net/publication/235090427_A_Statistical_Test_Suite_for_Random_and_Pseudorandom_Number_Generators_for_Cryptographic_Applications (accessed on 12 August 2024).
19. John, W. A Pseudorandom Number Sequence Test Program. 2008. Available online: <https://www.fourmilab.ch/random/> (accessed on 15 August 2024).
20. Wallace, K.; Moran, K.; Novak, E.; Zhou, G.; Sun, K. Toward sensor-based random number generation for mobile and IoT devices. *IEEE Internet Things J.* **2016**, *3*, 1189–1201. [CrossRef]
21. Kumar, V.; Rayappan, J.B.B.; Amirtharajan, R.; Praveenkumar, P. Quantum true random number generation on IBM's cloud platform. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 6453–6465. [CrossRef]
22. Valle, J.; Machicao, J.; Bruno, O.M. Chaotical PRNG based on composition of logistic and tent maps using deep-zoom. *Chaos Solitons Fractals* **2022**, *161*, 112296. [CrossRef]
23. Al-Mhadawi, M.M.; Albahrani, E.A.; Lafta, S.H. Efficient and secure chaotic PRNG for color image encryption. *Microprocess. Microsyst.* **2023**, *101*, 104911. [CrossRef]
24. Zhao, L.; Liao, X.; Xiao, D.; Xiang, T.; Zhou, Q.; Duan, S. True random number generation from mobile telephone photo based on chaotic cryptography. *Chaos Solitons Fractals* **2009**, *42*, 1692–1699. [CrossRef]
25. Yeoh, W.Z.; Teh, J.S.; Chern, H.R. A parallelizable chaos-based true random number generator based on mobile device cameras for the android platform. *Multimed. Tools Appl.* **2019**, *78*, 15929–15949. [CrossRef]
26. Truong, N.D.; Haw, J.Y.; Assad, S.M.; Lam, P.K.; Kavehei, O. Machine learning cryptanalysis of a quantum random number generator. *IEEE Trans. Inf. Forensics Secur.* **2018**, *14*, 403–414. [CrossRef]
27. De Bernardi, M.; Khouzani, M.H.R.; Malacaria, P. Pseudo-random number generation using generative adversarial networks. In *ECML PKDD 2018 Workshops, Proceedings of the Nemesis 2018, UrbReas 2018, SoGood 2018, IWAISe 2018, and Green Data Mining 2018, Dublin, Ireland, 10–14 September 2018*; Proceedings 18; Springer: Berlin/Heidelberg, Germany, 2018; pp 191–200. [CrossRef]
28. Ji, P.; Ma, H.; Ma, Q.; Chen, X. A novel method to generate pseudo-random sequence based on GAN. *J. Netw. Intell.* **2022**, *7*, 222–230. Available online : https://www.researchgate.net/publication/359176212_A_Novel_Method_to_Generate_Pseudo-Random_Sequence_based_on_GAN (accessed on 1 May 2024).
29. Man, Z.; Li, J.; Di X.; Liu, X.; Zhou, J.; Wang, J.; Zhang, X. A novel image encryption algorithm based on least squares generative adversarial network random number generator. *Multimed. Tools Appl.* **2021**, *80*, 27445–27469. [CrossRef]
30. Wu, Y.; Noonan, J.P.; Agaian, S. NPCR and UACI randomness tests for image encryption. *Cyber J. Multidiscip. J. Sci. Technol. J. Sel. Areas Telecommun. JSAT* **2011**, *1*, 31–38. Available online : <https://api.semanticscholar.org/CorpusID:8015168> (accessed on 10 May 2024).
31. Li, J.; Liu, Z.; Man, Z. Key generation method based on generative adversarial network and its application in low-light-level image encryption. *Acta Armamentarii* **2022**, *43*, 337–344. [CrossRef]
32. Saad, M.M.; Rehmani, M.H.; O'Reilly, R. Early stopping criteria for training generative adversarial networks in biomedical imaging. *arXiv* **2024**, arXiv:2405.20987. Available online: <https://arxiv.org/abs/2405.20987> (accessed on 27 December 2024).
33. Shi, Q.; Yang, Z.; Cheng, T.; Wang, C.; Wu, Z.; Zhang, X.; Xu, P. QKBAKA: A quantum-key-based authentication and key agreement scheme for the internet of vehicles. *IEEE Internet Things J.* **2023**, *11*, 12292–12306. [CrossRef]
34. Kim, H.; Kwon, Y.; Sim, M.; Lim, S.; Seo, H. Generative adversarial networks-based pseudo-random number generator for embedded processors. In *Information Security and Cryptology—ICISC 2020, Proceedings of the 23rd International Conference, Seoul, Republic of Korea, 2–4 December 2020*; Proceedings 23; Springer: Berlin/Heidelberg, Germany, 2021; pp. 215–234. [CrossRef]
35. Bonomi, F.; Mito, R.; Zhu, J.; Addepalli, S. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC Workshop on Mobile Cloud Computing, Helsinki, Finland, 17 August 2012*; pp. 13–16. [CrossRef]
36. Wazid, M.; Bagga, P.; Das, A.K.; Shetty, S.; Rodrigues, J.J.; Park, Y. AKM-IoV: Authenticated key management protocol in fog computing-based internet of vehicles deployment. *IEEE Internet Things J.* **2019**, *6*, 8804–8817. [CrossRef]
37. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for autonomous driving? The KITTI Vision Benchmark Suite. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012*; pp. 3354–3361. [CrossRef]
38. Moysis, L.; Volos, C.; Jafari, S.; Munoz-Pacheco, J.M.; Kengne, J.; Rajagopal, K.; Stouboulos, I. Modification of the logistic map using fuzzy numbers with application to pseudorandom number generation and image encryption. *Entropy* **2020**, *22*, 474. [CrossRef]

39. Yu, F.; Chen, H.; Wang, X.; Xian, W.; Chen, Y.; Liu, F.; Madhavan, V.; Darrell, T. BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 2633–2642. [[CrossRef](#)]
40. Zhu, Z.; Liang, D.; Zhang, S.; Huang, X.; Li, B.; Hu, S. Traffic-Sign Detection and Classification in the Wild. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
41. Ye, G.; Huang, X. A secure image encryption algorithm based on chaotic maps and SHA-3. *Secur. Commun. Netw.* **2016**, *9*, 2015–2023. [[CrossRef](#)]
42. Yang, H.; Wong, K.-W.; Liao, X.; Zhang, W.; Wei, P. A fast image encryption and authentication scheme based on chaotic maps. *Commun. Nonlinear Sci. Numer. Simul.* **2010**, *15*, 3507–3517. [[CrossRef](#)]
43. Luo, J.; Yu, W. An image encryption method based on random number matrix iterations. In Proceedings of the 2022 15th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Beijing, China, 5–7 November 2022; pp. 1–4. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.