sensors

MDPI

Article

A Mathematically Generated Noise Technique for Ultrasound Systems

Hojong Choi [1] and Seung-Hyeok Shin [2,*]

[1] Department of Electronic Engineering, Gachon University, Seongnam 13120, Republic of Korea
[2] Department of Mathematics and Big-Data Science, Kumoh National Institute of Technology, Gumi 39177, Republic of Korea
* Correspondence: shinbaad@kumoh.ac.kr

Abstract: Ultrasound systems have been widely used for consultation; however, they are susceptible to cyberattacks. Such ultrasound systems use random bits to protect patient information, which is vital to the stability of information-protecting systems used in ultrasound machines. The stability of the random bit must satisfy its unpredictability. To create a random bit, noise generated in hardware is typically used; however, extracting sufficient noise from systems is challenging when resources are limited. There are various methods for generating noises but most of these studies are based on hardware. Compared with hardware-based methods, software-based methods can be easily accessed by the software developer; therefore, we applied a mathematically generated noise function to generate random bits for ultrasound systems. Herein, we compared the performance of random bits using a newly proposed mathematical function and using the frequency of the central processing unit of the hardware. Random bits are generated using a raw bitmap image measuring 1000 × 663 bytes. The generated random bit analyzes the sampling data in generation time units as time-series data and then verifies the mean, median, and mode. To further apply the random bit in an ultrasound system, the image is randomized by applying exclusive mixing to a 1000 × 663 ultrasound phantom image; subsequently, the comparison and analysis of statistical data processing using hardware noise and the proposed algorithm were provided. The peak signal-to-noise ratio and mean square error of the images are compared to evaluate their quality. As a result of the test, the min entropy estimate (estimated value) was 7.156616/8 bit in the proposed study, which indicated a performance superior to that of GetSystemTime. These results show that the proposed algorithm outperforms the conventional method used in ultrasound systems.

Keywords: mathematically generated noise; ultrasound system; mathematical function

check for updates

Citation: Choi, H.; Shin, S.-H. A Mathematically Generated Noise Technique for Ultrasound Systems. *Sensors* **2022**, *22*, 9709. https://doi.org/10.3390/s22249709

Academic Editors: Diego Martín, Masoud Kaveh and Mohammad Reza Mosavi

Received: 16 October 2022
Accepted: 29 November 2022
Published: 11 December 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 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

Ultrasound systems can allow physicians to detect diseases for diagnostic purposes [1]; therefore, they can be used to determine the physiological conditions [2]. Ultrasound systems allow physicians to perform immediate treatment in outdoor sports games or island villages [3]. In particular, ultrasound systems have been developed owing to progress in application-specific integrated circuit technology [4,5]; therefore, they have been used to treat patients in island villages [6].

Ultrasound systems comprise access devices and these allow consultation by the physician [7]. Ultrasound is easier to perform without contact; however, ultrasound is currently performed using diagnostic machines, which are susceptible to cyberattacks [8]. Hardware and software development, including encryption algorithms, have been widely applied in automated teller machines and for internet access [9]. Compared with encrypting methods based on hardware, software-based algorithms are easily changeable and can be accessed by a software developer [10]; hence, software-based algorithms for supporting ultrasound have been developed, as shown in Figure 1. In our proposed concept, attacks must be controlled in the terminal machine when necessary.

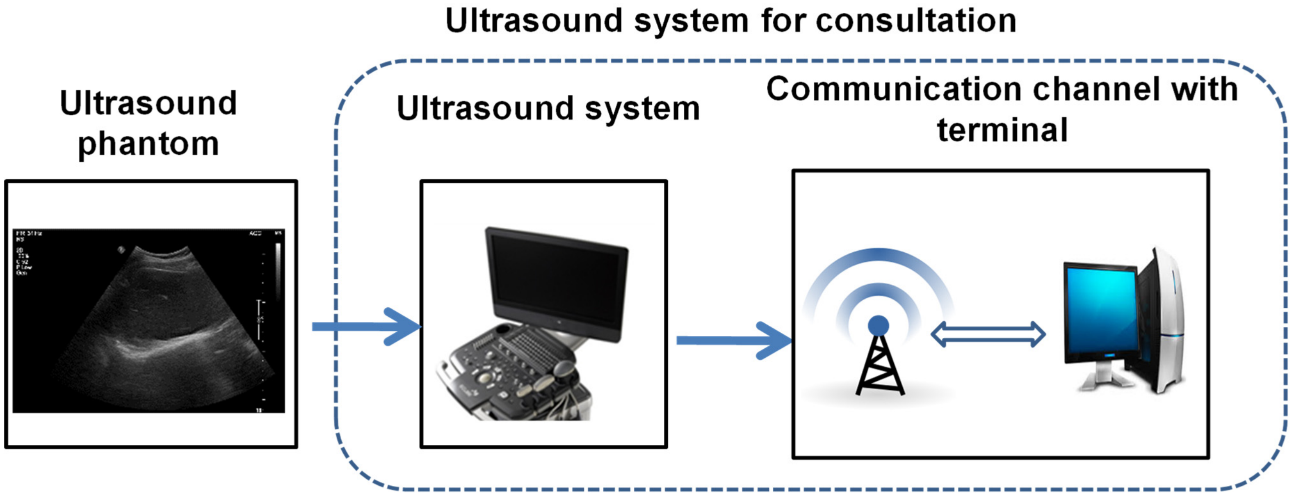

Ultrasound system for consultation
Ultrasound phantom
Ultrasound system
Communication channel with terminal

Figure 1. Our proposed concept of an ultrasound system for consultation.

Our proposed method can be utilized in conventional ultrasound machines since the latter feature 100–240 V alternating-current cords [7,11]; therefore, the encryption algorithm can easily be implemented at the final terminal. Most conventional ultrasound machines are based on outdated Microsoft Windows operating systems [12].

Owing to the development of information technology, awareness regarding the importance of information security and encryption technology has increased [13,14]. Encryption technology must satisfy the unpredictable stability of encrypted information protection application systems [15]. Random bits are basic elements in cryptographic algorithm design; therefore, they are vital to the construction of cryptographic systems [16]. These random bits are generated using cryptographic random-bit generators; therefore, they are used to ensure the stability of cryptographic and related application systems. The random-bit generators are classified into pseudorandom-bit generators (PRNGs) and true random-bit generators (TRNGs).

A cryptographically safe random-bit generator comprises a TRNG and a PRNG so it obtains the output value from the noise source as the input value of the TRNG and uses it as the seed for the PRNG to generate cryptographically safe random bits [17]. Conventional noise sources use noise generated by hardware, such as time, central processing unit (CPU) frequency, hard disk drive revolutions per minute, and memory usage [18]; however, extracting sufficient noise sources from systems is difficult when hardware resources are limited [19]; therefore, we propose a new method to generate a noise source using a mathematical algorithm and verify its performance by comparing it with hardware noise.

Two-post modules in hardware-based number generators were implemented to improve the randomness of linear PRNGs [20]. A chaos-based random number generator in a field-programmable gate array (FPGA) was implemented to achieve higher statistical capability in hardware [21]. Recent random-bit generation techniques for medical imaging applications between years 2018 and 2022 are summarized here. Stochastic gradient descent combined with the random number algorithm was implemented using a private deep learning network for medical imaging applications [22]. A random number was used to generate the weight of a deep learning mechanism for medical imaging [23]. A TRNG-based multilevel fusion technique was developed to improve cryptographic strength for medical imaging applications [24]. A TRNG-based AES encryption algorithm was proposed to develop robust medical systems [25].

Even if power is not supplied in the security field and major information such as hardware chip identification or security key is not lost, the same value should always be maintained when power is supplied again [26,27]; for this purpose, it is mainly stored in a non-volatile memory (NVM) such as an electrically erasable programmable read-only memory [28]; however, in the event of loss or theft of NVM, there is a risk of leakage of important confidential information [29]. It is also exposed to the risk of various physical attacks. One of the ways to solve this problem is the physical uncollectible function (PUF) technology, which generates unpredictable random numbers using process deviations that occur in semiconductor manufacturing processes [30].

The PUF generated from TRNG in the hardware is described. A latency-based dynamic random-access memory PUF without additional error correction technique was utilized for cryptography and authentication service areas [31]. A memory architecture using multi-bit entropy was used for hardware security applications [32]. The combinational TRNG and PUF techniques were designed for resistive random-access memory and FPGA applications [33,34]. The PUF was used for providing a cryptographic key to optimize the elements in the configurable logic block of the FPGA board [35].

Ultrasound is highly susceptible to cyberattacks [36]. In most of the current research, there are various methods for generating noises but most of these studies are based on hardware [37]. Compared with hardware-based encrypting algorithms, software-based algorithms can be updated with software engineers to increase encrypting capability [38]; therefore, we applied a mathematically generated noise function to generate random bits for ultrasound systems.

Section 2 describes the typical noise source generation methods using the CPU of the Microsoft Windows operating system and fundamental information regarding noise sources. Section 3 describes a method to generate random bits using hardware noise and presents the results of random-bit generation and randomized ultrasound images. Section 4 describes a method to derive mathematical functions through a noise generation algorithm for random-bit generation and, then, presents the results of random-bit generation and randomized ultrasound images. Section 5 provides a comparison and an analysis of statistical data processing using hardware noise and the proposed algorithm. Section 6 presents the conclusion and future work of the designed noise generation algorithm.

2. Materials and Methods

A method for generating random bits using mathematical functions is proposed herein. To compare and verify this method, hardware noise sources provided by Microsoft's Windows system were compared with those provided by the proposed algorithm. The hardware noise source depends on the operating environment and operating system, and they tend to exhibit a uniform distribution and low entropy [39]. In this study, the sampling intervals were diversified based on time for the mass generation of random bits. The time-generated time-series data and random-bit generation distributions were extracted and compared with the designed algorithm to evaluate their randomness. In the experiment, a 1000×663 bitmap image was randomized with the generated random bit to validate the random bit after randomizing the image.

The random-bit generator generates random bits via the noise-source acquisition, entropy accumulation and seed generation, and random-bit heat generation steps [40]. The output random bit depends on the input noise source which can be evaluated using both hardware and software noise sources [41]. Table 1 shows the noise sources associated with the CPU frequency and CPU elapsed time among the noise sources available in the Microsoft Windows operating system. In the document NIST SP 800-90B, a physical source dependent on hardware and a non-physical noise source using system data are defined [42]; thus, we selected local time and a non-physical noise source to compare performance as a noise source generated by software.

Table 1. Noise source associated with the CPU of the Microsoft Windows operating system.

Noise Source	Sample Size (Byte)
GetTickCount	4
GetSystemTime	16
QueryPerformanceCounter	64

Among the noise sources based on time, GetTickCount and QueryPerformanceCounter use the clock and frequency of the CPU, whereas GetSystemTime uses the time (in microseconds) provided by the system. The GetTickCount, GetSystemTime, and QueryPerformanceCounter from windows API are functions implemented in Microsoft's C++ language. First, GetTickCount returns the delayed time to the function call in msec from the time the system starts [43]. Second, GetSystemTime returns the current time in Universal Time format [44]. Third, QueryPerformanceCounter returns the current frequency [45]. GetSystemTime uses time-series data and can be created as expressed as shown in Equation (1).

$$t_n = (t_1 + dt) \bmod 256, \tag{1}$$

where t_1 is the initial time and dt is the amount of change in time.

In Equation (1), mod 256 is a modulo operation that uses 8 bits of the least sequence bit (LSB). A random bit based on time may exhibit higher entropy as the value of dt increases. This implies that random bits require a long time to generate high entropy. QueryPerformanceCounter is a method for measuring the CPU frequency. In Equation (2), frequency is expressed as a random bit based on the initial frequency and elapsed time.

$$f_n = (f_1 + dt) \bmod 256, \tag{2}$$

where f_1 is the initial frequency and dt is the amount of change in time.

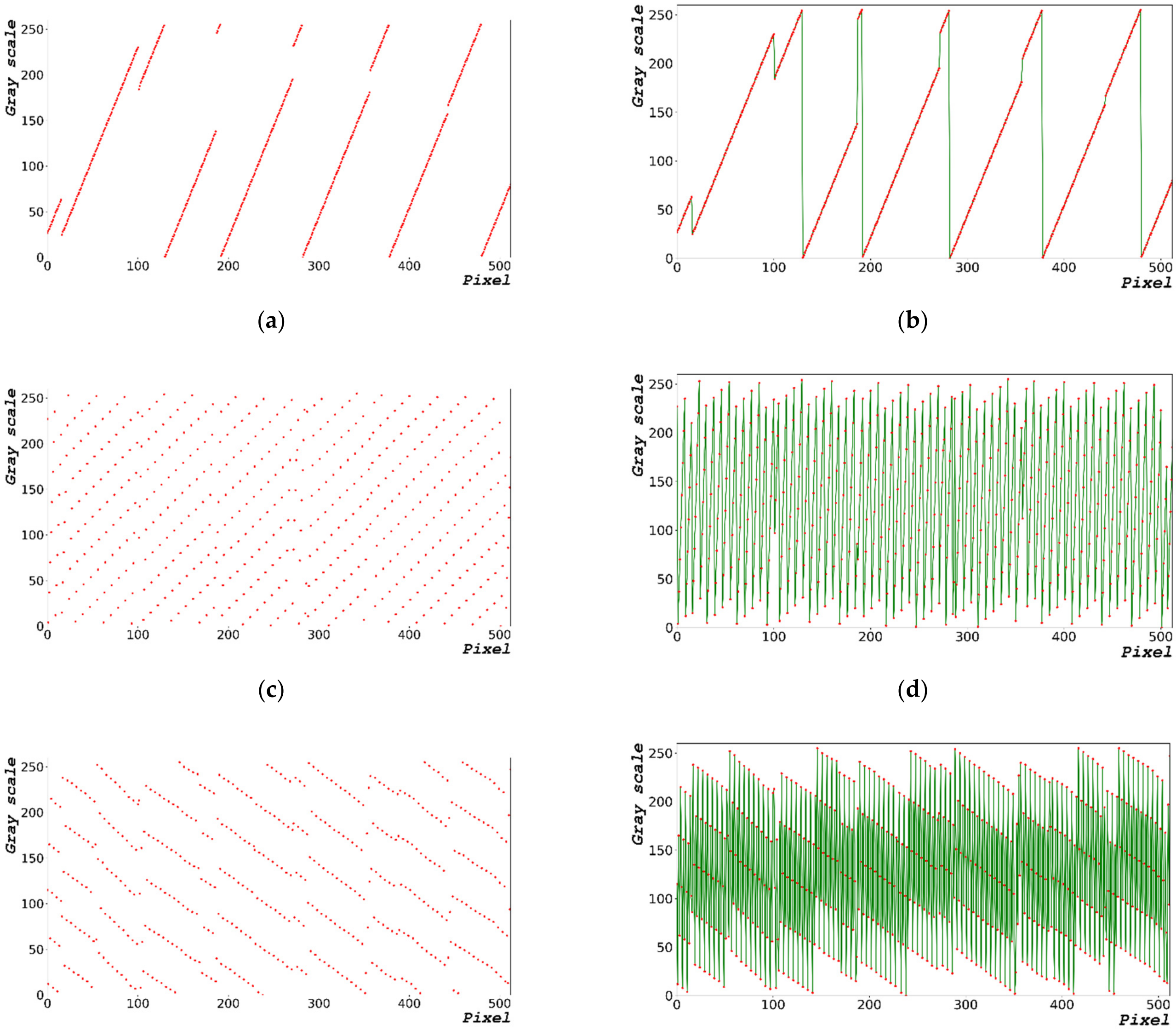
Table 2 lists the frequency counts obtained after the time elapsed by dt. The CPU frequency exhibits a higher rate of change in the 8-bit LSB than in a 24-bit most sequence bit (MSB); therefore, the entropy of the random bit based on the CPU frequency was determined to be 8 bits. As shown in Equation (2), the result of the modulo operation at 256 is combined to generate a random bit of the desired size. The size of the noise source can be generated as 4–64 bytes, but the lower one byte is used as entropy because overlapping values appear in the upper byte.

Table 2. Frequency of elapsed time.

Elapsed Time	Frequency
	MSB LSB
:	:
$f_1 + kd$	5A561A4C
$f_1 + (k+1)d$	5A561D9A
$f_1 + (k+2)d$	5A56200D
:	5A5622E6
:	5A5625CA
:	5A562878
	5A562AE7
	5A562D66
	5A563031
	5A5632FD
	5A56357F
	5A5637DF
$f_1 + nd$	5A563A7F
:	:

3. Random-Bit Generation by Hardware Noise

Figure 2 shows the process of generating random bits with 1000 × 663 pixels. Equation (2) expresses the equation to generate some of the time-series data (0 to 512 pixels) of the random-bit distribution diagram. The pixels of the time-series data on the x-axis represent the flow of time. Figure 2a,c,e show the random-bit generation time-series data at dt = 0, 1.0, and 5.0 µs, respectively. Figure 2b,d,f show the linear trend line representing the distribution and time flow of Figure 2a,e, respectively; in particular, they present the distribution of random-bit generation between 0 and 100 pixels with a linear trend line added. The closer dt is to zero, the higher is the number of random bits that are generated linearly. This implies that the generation of random bits can be predicted. The larger the dt, the greater is the entropy of the random bit; however, the increase in dt significantly increases the overall random-bit generation time.



(**a**) (**b**)

(**c**) (**d**)

(**e**) (**f**)

Figure 2. Random-bit generation result based on time: (**a**) 1000 × 663 random-bit generation time-series data; (**b**) generated random-bit trend line for dt = 0 µs; (**c**) 1000 × 663 random-bit generation time-series data; (**d**) generated random-bit trend line for dt = 1.0 µs; (**e**) 1000 × 663 random-bit generation time-series data; and (**f**) generated random-bit trend line for dt = 5.0 µs. Red dot and green line represent the distribution and time flow of the random-bit, respectively.

Table 3 shows the mean, median, and mode of the generated random bit shown in Figure 2. The results are those of a random bit generated in the range of 0 to 255 pixels in a 1000 × 663 image. The generated average was 127 and the median value was 128, which resulted in an even distribution. The trend lines in Figure 2b,d,f show a disadvantage, i.e., the distribution of the generated random bits is linear.

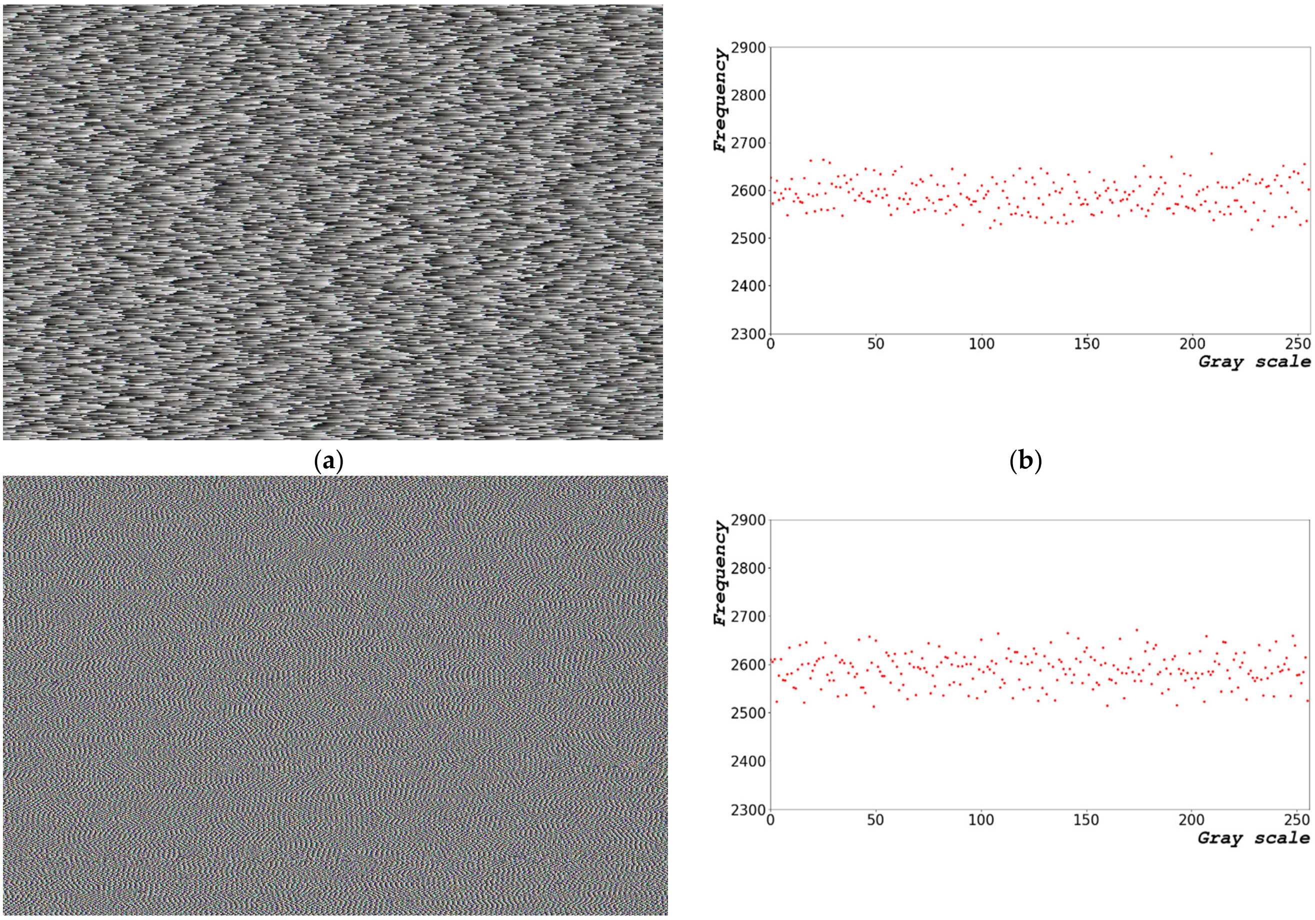
Table 3. Statistics data of Figure 2.

dt	Mean	Median	Mode
0 µs	127.43	127	209
1.0 µs	127.50	128	174
5.0 µs	127.53	128	181

Figure 3 shows the random bits generated using Equation (2). Figure 3a shows an image of the random bit generated and Figure 3b presents the distribution of the 8-bit random bits shown in Figure 3a. The random-bit distribution shaped by the grayscale (Figure 3a) exhibits a constant pattern. In Figure 3a, the overall image form is repeated from dark to bright as it progresses from left to right. Figure 3a,c,e show the 1000 × 663 random-bit visualization images at dt = 0, 1.0, and 5.0 µs, respectively. Figure 3b,d,e show the 0–255, 8-bit entropy generation distribution diagram for dt = 0, 1.0, and 5.0 µs, respectively.

The results shown in Figure 3 indicate that the pattern of random bits generated over time appears as a repeating pattern of constant values from left to right. When the random bit is generated using the frequency counter of the CPU, a certain pattern is yielded, which is disadvantageous.

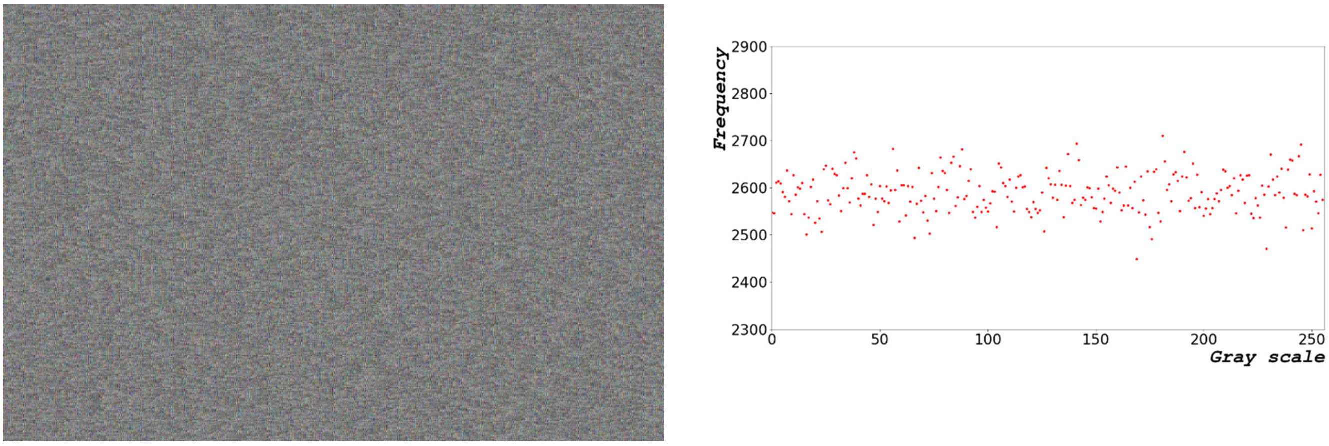
We attempted to verify the randomness of the noise source by connecting random number candidates of up to 8 bits to generate a noise source of 1024 bits or more. Therefore, we obtain the images using an 8 bit value corresponding to 1 byte so the grayscale range is 0 to 255. The image data of the ultrasound phantom were obtained using an ultrasound machine.



(**a**) (**b**)

(**c**) (**d**)

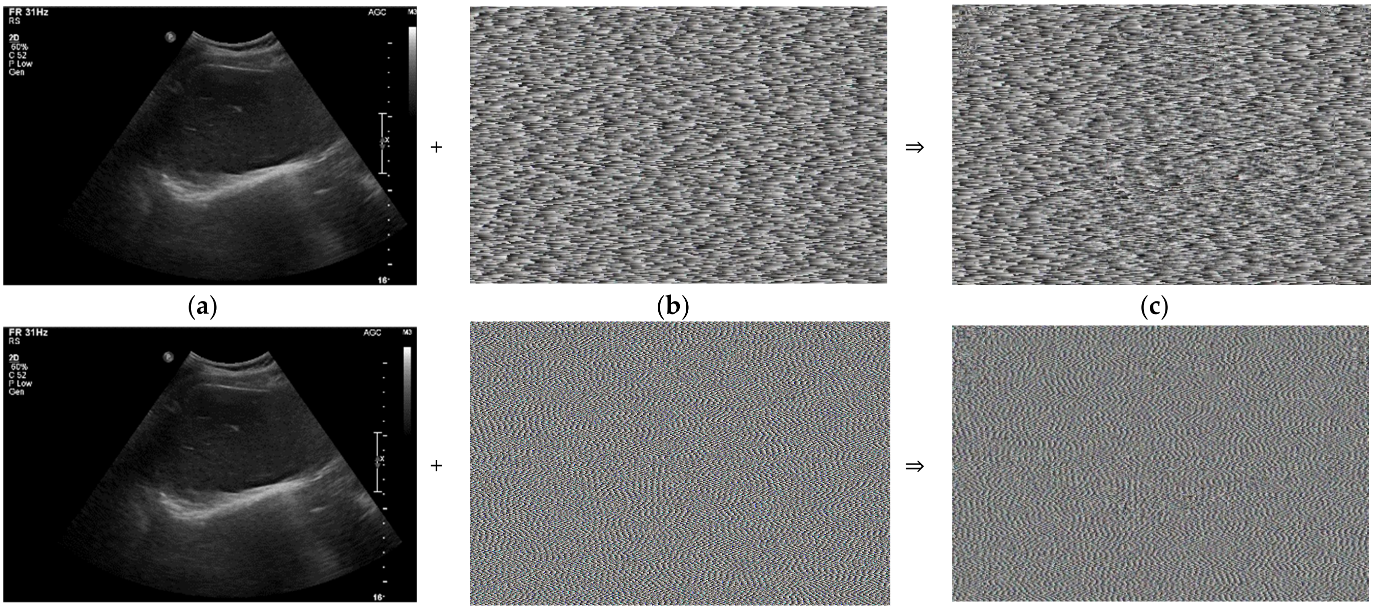
Figure 3. *Cont.*



(**e**) (**f**)

Figure 3. Random-bit generation results based on time: (**a**) 1000 × 663 random-bit visualization image; (**b**) 0–255, 8-bit entropy generation distribution diagram at dt = 0 µs; (**c**) 1000 × 663 random-bit visualization image; (**d**) 0–255, 8-bit entropy generation distribution diagram at dt = 1.0 µs; (**e**) 1000 × 663 random-bit visualization image; and (**f**) 0–255, 8-bit entropy generation distribution diagram at dt = 5.0 µs.

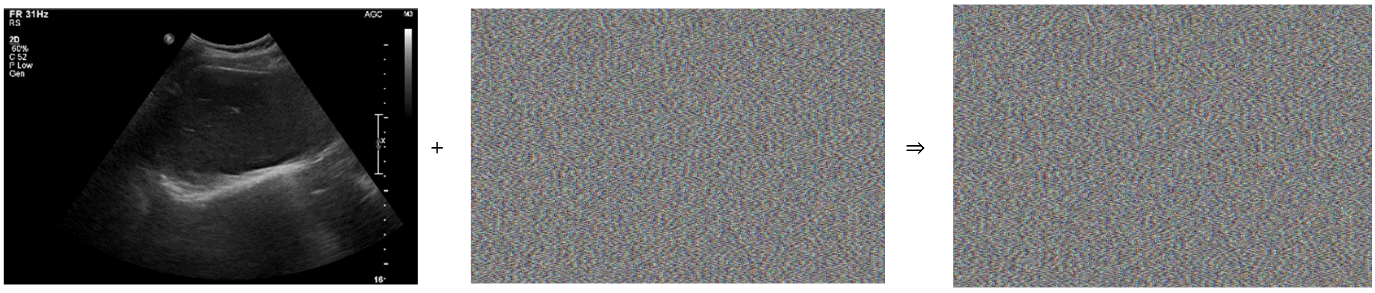
Figure 4a,d,g show the same ultrasound abdomen phantom images captured from the ultrasound machine. Figure 4b,e,h show the 1000 × 663 random-bit generation visualization image at dt = 0, 1.0, and 5.0 µs, respectively. Figure 4c,f,i show the randomized ultrasound images combined from the ultrasound image and random-bit generation visualization image at dt = 0, 1.0, and 5.0 µs, respectively.



(**a**) (**b**) (**c**)

(**d**) (**e**) (**f**)

Figure 4. *Cont.*



(**g**) (**h**) (**i**)

Figure 4. Randomized ultrasound abdomen phantom images: (**a**) ultrasound image; (**b**) 1000 × 663 random-bit generation visualization image; (**c**) randomized ultrasound image at dt = 0.5 µs; (**d**) ultrasound image; (**e**) 1000 × 663 random-bit generation visualization image; (**f**) randomized ultrasound image at dt = 1 µs; (**g**) ultrasound image; (**h**) 1000 × 663 random-bit generation visualization image; and (**i**) randomized ultrasound image at dt = 5.0 µs.

4. Random-Bit Generation via Designed Noise Generation Algorithm

The random bit is a true random bit if the probability functions Pr for both X and Y variables are not related; thus, we can obtain the following relationship [46]:

$$\Pr\,(X \cup Y) = \Pr(X)\Pr(Y) \tag{3}$$

In general, a true random bit cannot be generated or identified; therefore, the random bit is assumed to be deterministic. The deterministic random bit must satisfy a one-to-one corresponding function as shown in Equation (4) [47].

$$x_k \neq x_{k+1} \Leftrightarrow f(x_k) \neq f(x_{k+1}) \tag{4}$$

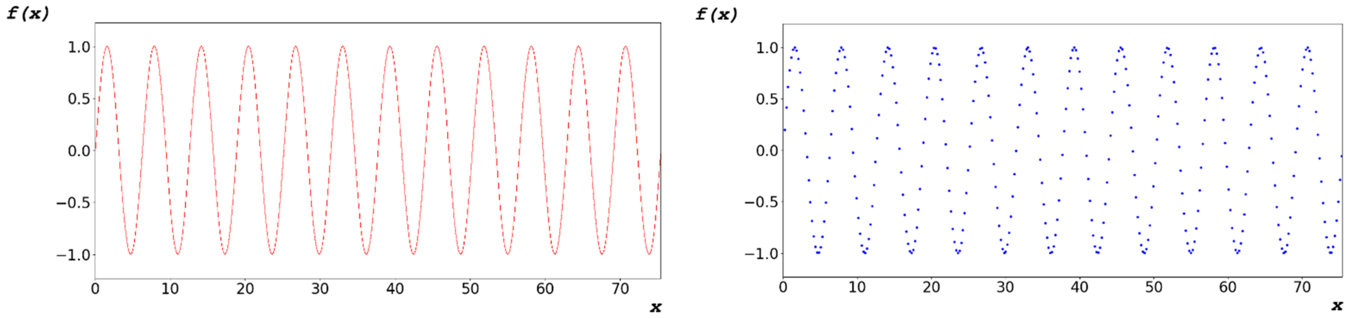
where $k \in N$.

As shown in Figure 5a, the periodic function vibrates at regular intervals p, as described in Equation (5). This function does not reflect a one-to-one correspondence, as described in Equation (6).

$$f(x_k) = f(x_k + p) \tag{5}$$

$$\begin{gathered} x_k \neq x_{k+1} \Rightarrow f(x_k) = f(x_{k+1}) \\ \text{Let } x_{k+1} = x_k + p \Rightarrow f(x_k) = f(x_{k+1}) \Rightarrow f(x_k) = f(x_k + p) \end{gathered} \tag{6}$$

where $k \in N$ and $p \in R$

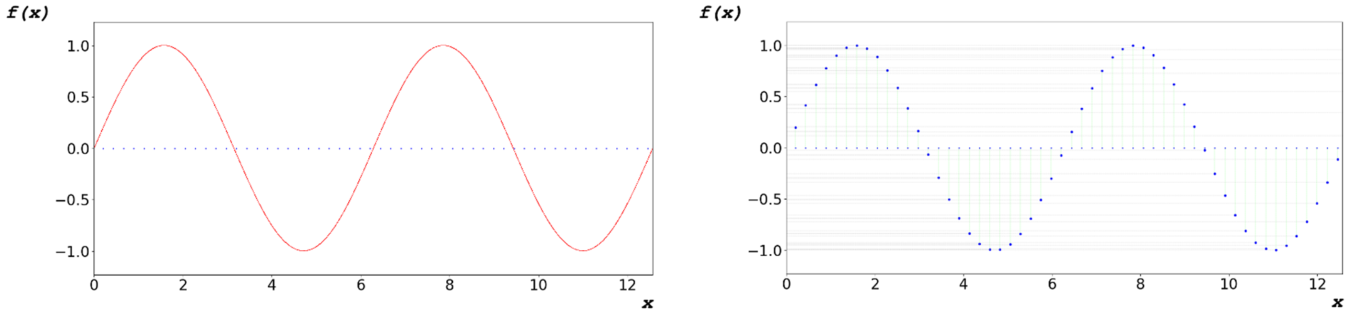


(**a**) (**b**)

Figure 5. (**a**) Mathematical continuous periodic function; and (**b**) discontinuous periodic function.

To generate the random bit of the algorithm, as shown in Figure 5b, the independent variable x of the domain was generated, as shown in Figure 6a, by adding the change dx of the first term a_1 and x, as described in Equation (7). The dependent variable y of the function covariate was generated as a random bit, as shown in Figure 6b. The dependent variable y generated using Equation (7) was converted to a floating-point hexadecimal number defined in IEEE754 [48].

$$\begin{gathered} a_1 = \text{current\_time( )} \\ x_n = a_1 + ndx, \text{ where } n \in Z \\ y = f(x) \end{gathered} \tag{7}$$



(**a**) (**b**)

Figure 6. (**a**) Determination of independent variables at regular intervals (dx) for generating random bits in periodic functions; and (**b**) set of non-overlapping dependent variables based on selected independent variables.

As shown in Figure 6, because the designed algorithm uses a periodic function, an irrational number cannot be used as a period. As described in Equation (8), the rounded value obtained is used at a certain point below the decimal point. The function with a period of rounded q produces an irrational number of q_r errors in the normal period, as indicated in Equation (9); therefore, a one-to-one correspondence exists, as in Equation (10), although q is a periodic function.

$$q = \lfloor p \times 10^n \rfloor \times 10^{-n}, \text{ where } n \in N \text{ and } p \in I, \tag{8}$$

$$q_r = p - q \tag{9}$$

where n is the index, N is a rational number, and I is an irrational number.

The proof for Equation (9) is presented below.

Theorem 1. $Q + I = I$
Here, Q is a rational number and I is an irrational number.

Proof. Assume that Q + I = Q
Let a, b, m, and n ∈ Z; subsequently, x ∈ I must be selected.
Furthermore, a, b, m, n, and Z are integer variables.

$$x + \frac{a}{b} = \frac{m}{n} \tag{10}$$

$$x + \frac{a}{b} - \frac{a}{b} = \frac{m}{n} - \frac{a}{b} \Rightarrow x = \frac{m}{n} - \frac{a}{b} = \frac{mb - na}{nb} \tag{11}$$

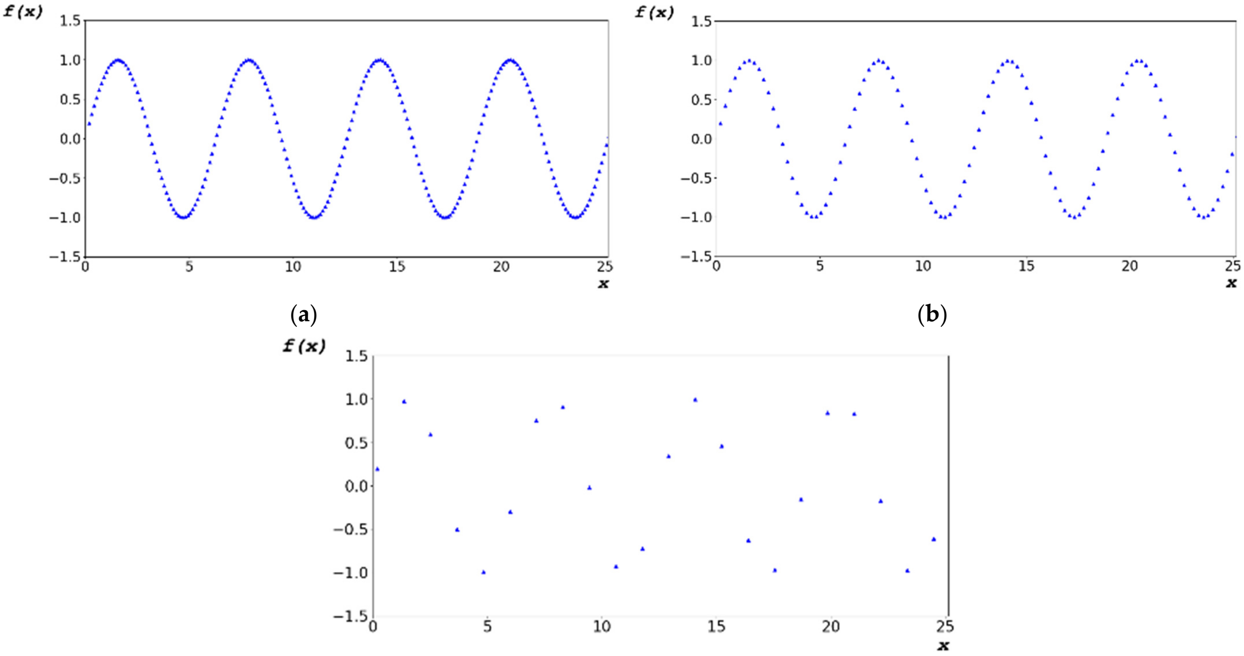
$$\Rightarrow x = \frac{mb - na}{nb} \tag{12}$$

Because variable x is a rational number, this proof is a contradiction; therefore, we confirm that x is an irrational number. □

The designed algorithm satisfies Equation (13) because it uses a periodic function that provides an irrational number. This periodic function is illustrated in Figure 5b.

$$\begin{gathered} x_k \neq x_{k+1} \Rightarrow f(x_k) \neq f(x_{k+1}) \\ f(x) \neq f(x+q) \end{gathered} \tag{13}$$

To test the designed algorithm, a random bit was generated by varying dx, as shown in Figure 7. Figure 7a–c show the periodic functions yielded using the designed algorithms at dx = 0.1156, 0.2312, and 1.1560 rad, respectively.



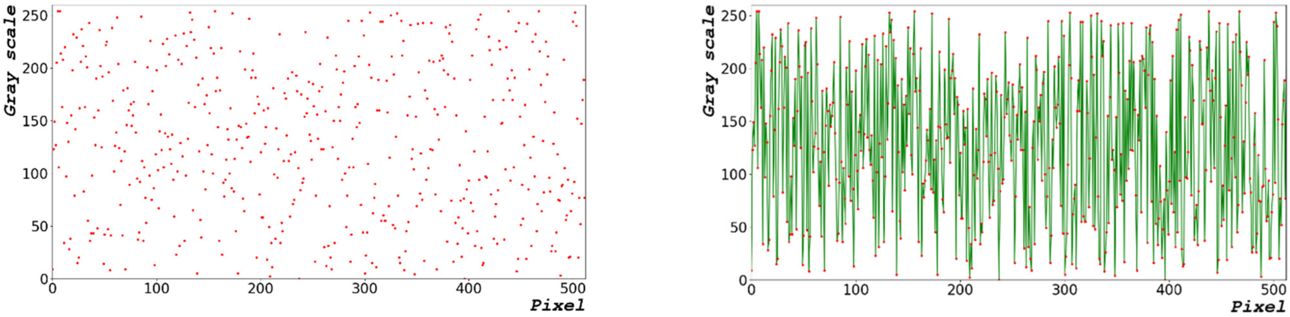
(**a**)

(**b**)

(**c**)

Figure 7. Periodic functions yielded using designed algorithms at (**a**) dx = 0.1156 rad, (**b**) dx = 0.2312 rad, and (**c**) dx = 1.1560 rad.

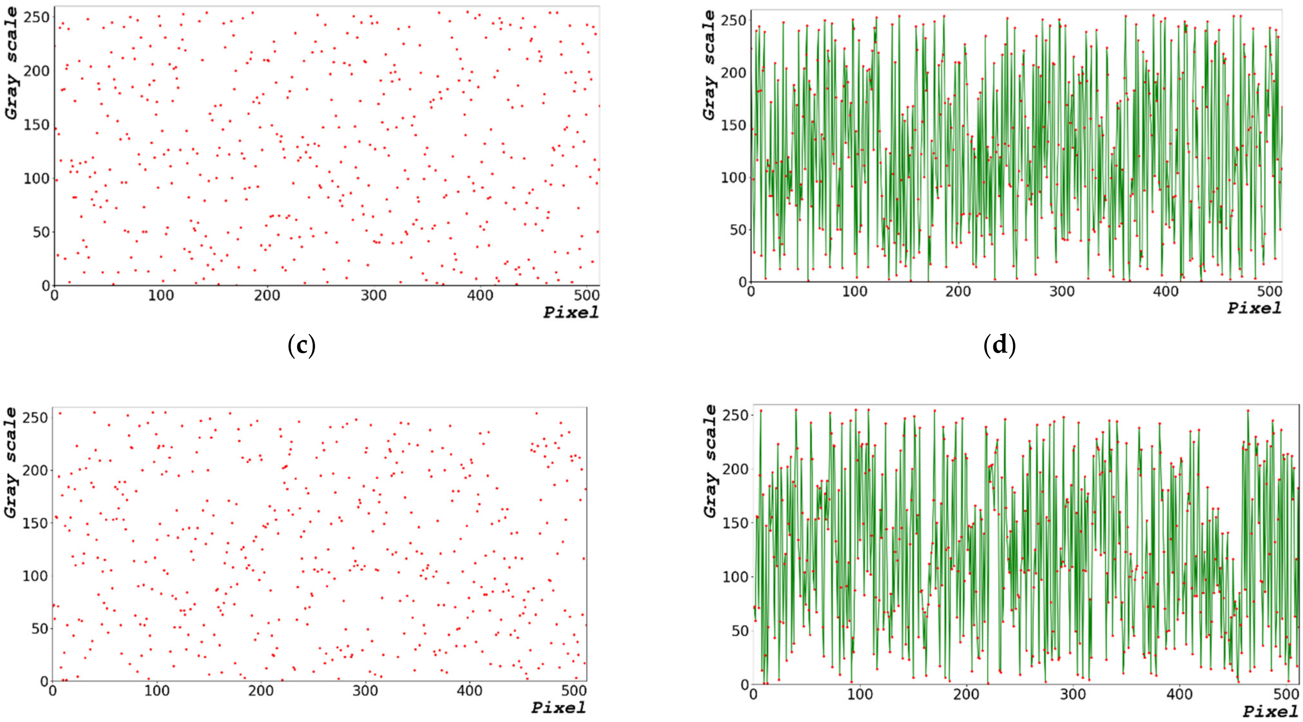
Figure 8 shows a graph of the random-bit distribution generated by 1000 × 663 images using the designed algorithm. Figure 8a shows the graph for dx = 0.1156 rad. Figure 8b shows the graph of Figure 8a with a trend line added for d_X = 0.1156 rad. Figure 8c,e show the graphs for dx = 0.2312 rad and 1.156 rad, respectively. Based on Figure 2, the data distribution was arbitrarily generated compared with that obtained using the CPU frequency provided by the hardware; in addition, the generation distribution was constant, despite the change in dx.



(**a**)

(**b**)

Figure 8. *Cont.*



(**c**) (**d**)

(**e**) (**f**)

Figure 8. Random-bit generation result in radian: (**a**) 1000 × 663 random-bit generation time-series data; (**b**) trend line of Figure 8a for d_X = 0.1156 rad; (**c**) 1000 × 663 random-bit generation time-series data; (**d**) trend line of Figure 8c for dx = 0.2312 rad; (**e**) 1000 × 663 random-bit generation time-series data; and (**f**) trend line of Figure 8c for dx = 1.1560 rad.

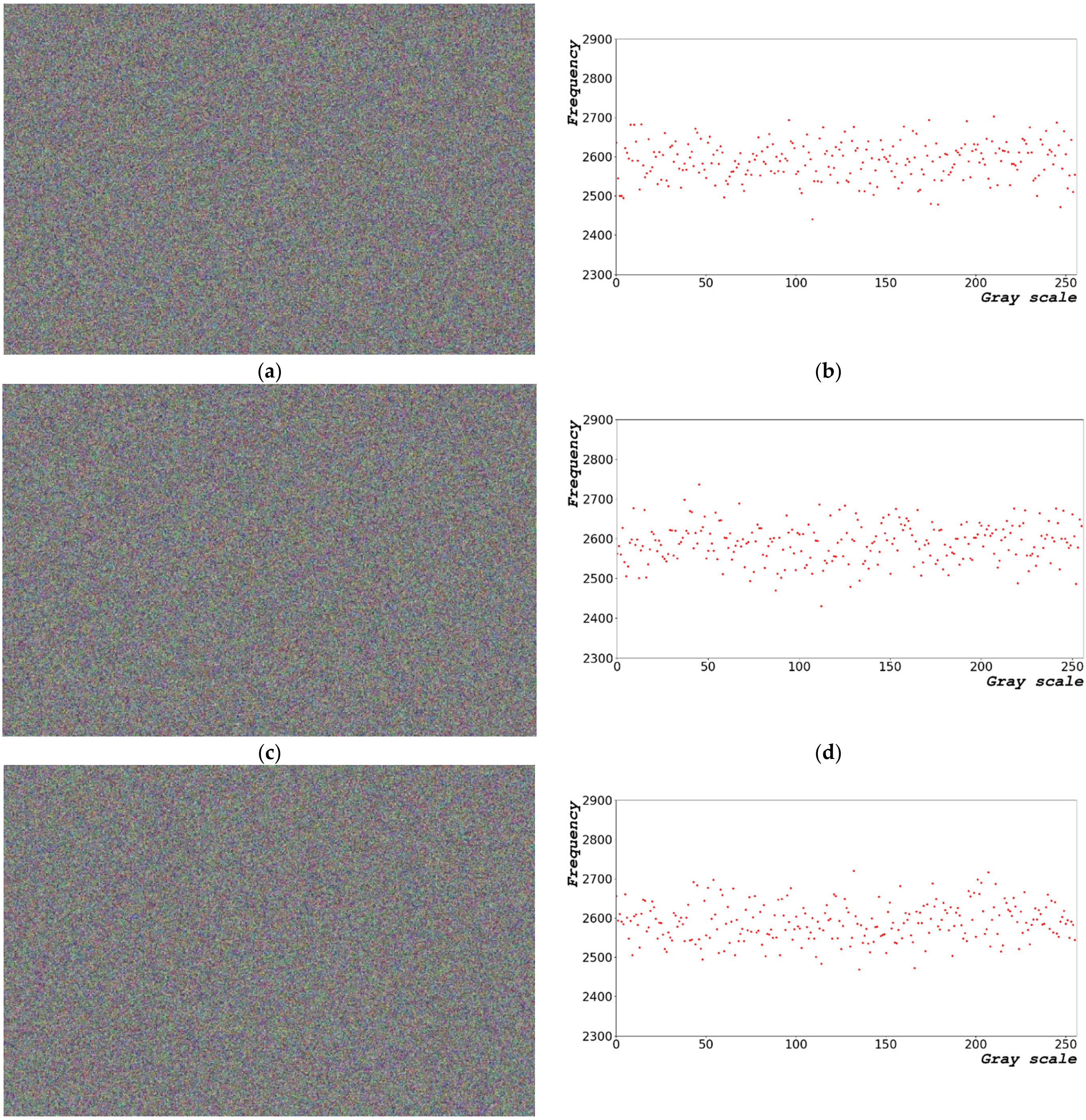
Table 4 lists the mean, median, and mode of the generated random bits shown in Figure 8. The results are those of a random bit generated in the range of 0 to 255 pixels in a 1000 × 663 image; the generated average was 127, and the median value was 128, which resulted in an even distribution.

Table 4. Statistics data of Figure 8.

dt	Mean	Median	Mode
0.1156 rad	127.56	128	210
0.2312 rad	127.59	128	45
1.1560 rad	127.62	128	132

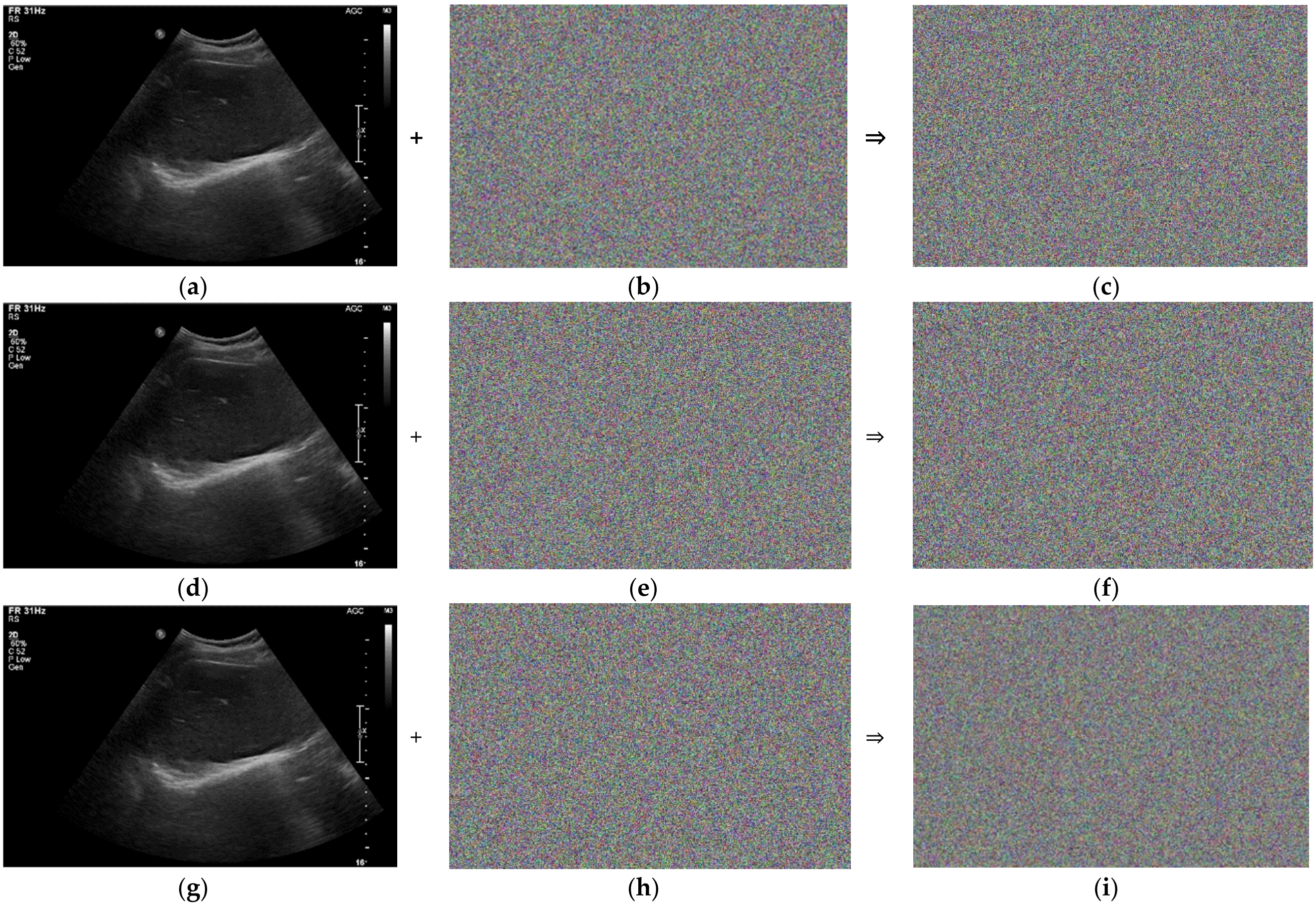
Figure 9a shows an image of the distribution of random bits generated by the designed algorithm. Figure 9b shows the distribution of the generated random bits. Figure 9a,c,e show the images of random-bit generation measuring 1000 × 663 at dx = 0.1156, 0.2312, and 1.1560 rad, respectively. Figure 9b,d,f show the distribution diagrams of entropy generation with sizes ranging from 0 to 255 and 8 bits at dx = 0.1156, 0.2312, and 1.1560 rad, respectively.

Figure 10a,d,g show the same ultrasound abdomen phantom images. Figure 10b,e,h show the 1000 × 663 random-bit generation visualization image at dx = 0.1156, 0.2312, and 1.1560 rad, respectively. Figure 10c,f,i show the randomized ultrasound images obtained by combining the ultrasound and random-bit generation visualization images at dx = 0.1156, 0.2312, and 1.1560 rad, respectively.



(**a**) (**b**) (**c**) (**d**) (**e**) (**f**)

Figure 9. Random-bit generation result based on time: (**a**) image of random-bit generation measuring 1000 × 663; (**b**) distribution diagram chart of entropy generation with size ranging from of 0 to 255 and 8 bits at dx = 0.1156 rad; (**c**) image of random-bit generation measuring 1000 × 663; (**d**) distribution diagram chart of entropy generation with size ranging from 0 to 255 and 8 bits at dx = 0.2312 rad; (**e**) image of random-bit generation measuring 1000 × 663; and (**f**) distribution diagram chart of entropy generation with size ranging from 0 to 255 and 8 bits at dx = 1.1560 rad.



(**a**) (**b**) (**c**)

(**d**) (**e**) (**f**)

(**g**) (**h**) (**i**)

Figure 10. Randomization of abdomen phantom image: (**a**) ultrasound image; (**b**) 1000 × 663 random-bit generation visualization image; (**c**) randomized ultrasound image for dx = 0.1156 rad; (**d**) ultrasound image; (**e**) 1000 × 663 random-bit generation visualization image; (**f**) randomized ultrasound image for dx = 0.2312 rad; (**g**) ultrasound image; (**h**) 1000 × 663 random-bit generation visualization image; and (**i**) randomized ultrasound image for dx = 1.156 rad.

5. Comparison Data of Ultrasound Image

Figure 11a–c show the enlarged graphs of Figure 2b,d,f, respectively. Figure 11d–f show the enlarged graphs of Figure 8b,d,f in the range of 0–128 pixels, respectively. The statistical data in Tables 3 and 4 show only the result; therefore, the distribution of random-bit generation cannot be verified. Based on Figure 11, the random-bit generation process can be analyzed using the random-bit generation trend line and distribution. Compared with the randomized data shown in Figure 11a–c, the randomized data as shown in Figure 11d–f are more complex.

Figure 12b,c show the enlarged images of Figures 4i and 10i, respectively. Figure 11b shows the randomization by the conventional hardware noise source, which allows the original image pattern to be maintained. Figure 12c shows the randomization by the designed algorithm, which does not maintain the pattern of the original image.

Table 5 lists the noise measurement results of the randomized images in Figure 11b,c based on the original image shown in Figure 11a. The mean square error (MSE) is a value representing the error between the reference and comparison images [49]. The peak signal-to-noise ratio (PSNR) is calculated using Equation (14) [50].

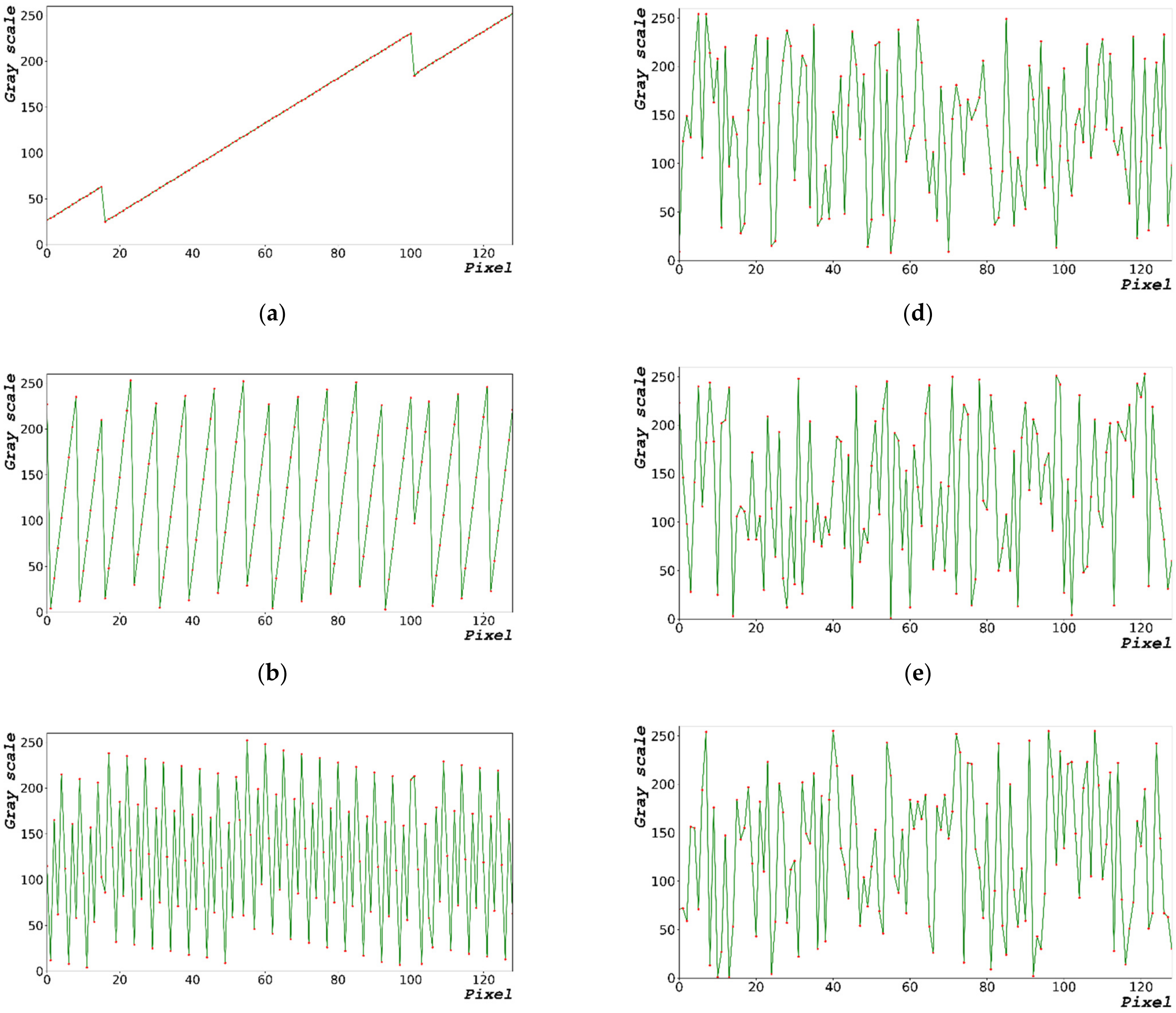
$$\mathrm{PSNR} = 10 \log \frac{\mathrm{s}^2}{\mathrm{MSE}}, \tag{14}$$

where s is the maximum value of the pixel.

Table 5 shows the comparison data of the image quality based on the conventional method using hardware noise and based on the proposed method using the designed algorithm. In the conventional method, random bits are generated by the hardware noise. In the proposed method, random bits are generated by software. The MSE is the difference in quality between the original and comparison images [51]; therefore, a high MSE value indicates a significant difference between the original and comparison images. As shown in Equation (14), the PSNR is a metric for evaluating an image and is inversely proportional to the MSE.

Table 5 shows the noisy images based on hardware noise sources (conventional) shown in Figure 4c,f,i, and those based on the proposed algorithms shown in Figure 10c,f,i (proposed). The MSE and PSNR values shown in Table 5 are extracted from Figure 4c,f,i, and Figure 10c,f,i, respectively. The MSE of the proposed algorithm is higher than that yielded by the method using hardware noise.

Table 6 shows the test result of the min entropy estimate (estimated value) vs. time. As a result of the test, the min entropy estimate (estimated value) was 7.156616/8 bit in the proposed study, which showed superior performance than GetSystemTime (Conventional).

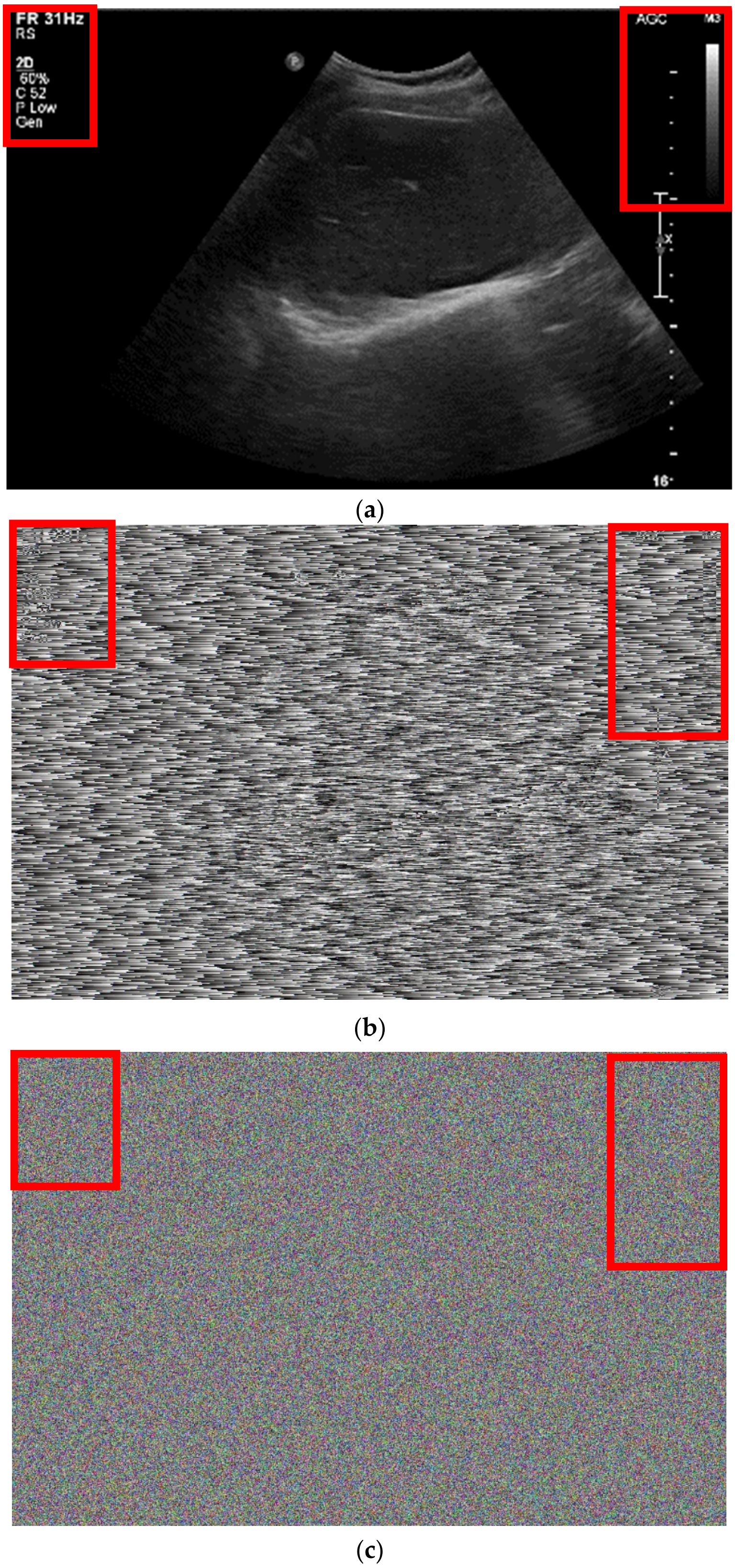


(**a**) (**d**)

(**b**) (**e**)

(**c**) (**f**)

Figure 11. Comparison data for encrypting performance. Enlarged sections of Figures (**a**) 2b, (**b**) 2d, and (**c**) 2f. Enlarged sections of Figures (**d**) 8b, (**e**) 8d, and (**f**) 8f within 0–128 pixels.



(**a**)

(**b**)

(**c**)

Figure 12. (**a**) Obtained original ultrasound image and enlarged images of (**b**) Figure 4i and (**c**) Figure 10i.

Table 5. Comparison data of image quality using conventional and proposed methods.

		Conventional			Proposed	
	0 μs	**1.0 μs**	**5.0 μs**	**0.1156 rad**	**0.2312 rad**	**1.1560 rad**
PSNR	5.593	5.594	5.594	5.587	5.588	5.589
MSE	17,937.932	17,930.966	17,933.858	17,959.493	17,955.940	17,953.607

Table 6. NIST SP 800-90B test results.

		Conventional			Proposed	
	0 μs	**1.0 μs**	**5.0 μs**	**0.1156 rad**	**0.2312 rad**	**1.1560 rad**
min entropy	0.1089	0.1232	0.1371	7.1566	7.1578	7.1603

6. Conclusions

Ultrasound systems have been widely used owing to their applicability. Ultrasound systems that can provide consultation are more likely to be hacked via cyberattacks; therefore, hardware- or software-based algorithms have been implemented to protect confidential patient information. Compared with hardware-based algorithms, software-based algorithms can be easily accessed by software engineers. In our proposed concept for ultrasound systems, an encryption algorithm is implemented in ultrasound machines via a terminal. Such ultrasound systems necessitate random bits to protect patient information. Unpredictable noise generated in the hardware is typically used to generate a random bit; however, extracting sufficient noise from systems is challenging when hardware resources are limited.

Herein, we proposed a new random-bit generation algorithm that can be used in ultrasound systems. Mathematically generated random bits are limited by the hardware memory size; however, the proposed algorithm is not constrained by the memory size because it uses values that diverge vibrations within a certain range using periodic functions. It can generate noise using a mathematical algorithm in a hardware-constrained system, such as an ultrasound system; subsequently, we verified its randomness performance via several experiments.

To confirm the feasibility of our proposed method, we extracted the images of an ultrasound phantom from an ultrasound system and then compared the image quality using the conventional and proposed methods. The MSE obtained when using the proposed algorithm (17,953.607) was higher than that when using hardware noise (17,933.858), and the PSNR when using the proposed algorithm (5.589) was lower than that when using hardware noise (5.594).; therefore, the randomized image obtained using the proposed algorithm indicated further deterioration in the image quality, which shows that the entropy for the random bit of the proposed algorithm is greater than that of the method that uses hardware noise for the ultrasound system. Our proposed method showed noise generation technique. In the future work, the developed technique will be applied to the embedded system and, then, generate actual key generation and use it as an entropy to increase encryption strength; in addition, research on efficiency in an actual system using the developed technique will be conducted.

Author Contributions: Conceptualization, S.-H.S. and H.C.; methodology, S.-H.S. and H.C.; formal analysis, S.-H.S. and H.C.; writing—original draft preparation, S.-H.S. and H.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Kumoh National Institute of Technology (2021).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are included in this article.

Conflicts of Interest: The authors declare no conflict 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. Shung, K.K.; Thieme, G.A. *Ultrasonic Scattering in Biological Tissues*; CRC Press: Boca Raton, FL, USA, 1992.
2. Huang, B.; Shung, K.K. Characterization of high-frequency, single-element focused transducers with wire target and hydrophone. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control* **2005**, *52*, 1608–1612. [CrossRef] [PubMed]
3. Duan, S.; Liu, L.; Chen, Y.; Yang, L.; Zhang, Y.; Wang, S.; Hao, L.; Zhang, L. A 5G-powered robot-assisted teleultrasound diagnostic system in an intensive care unit. *Crit. Care* **2021**, *25*, 134. [CrossRef] [PubMed]
4. Moore, C.L.; Copel, J.A. Point-of-care Ultrasonography. *N. Engl. J. Med.* **2011**, *364*, 749–757. [CrossRef] [PubMed]
5. Brunner, E. How ultrasound system considerations influence front-end component choice. *Analog Dialogue* **2002**, *36*, 1–4.
6. Kim, G.-D.; Yoon, C.; Kye, S.-B.; Lee, Y.; Kang, J.; Yoo, Y.; Song, T.-K. A single FPGA-based portable ultrasound imaging system for point-of-care applications. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control* **2012**, *59*, 1386–1394.
7. Daniels, J.M.; Hoppmann, R.A. *Practical Point-of-Care Medical Ultrasound*; Springer: New York, NJ, USA, 2016.
8. Adhikari, S.; Blaivas, M. *The Ultimate Guide to Point-of-Care Ultrasound-Guided Procedures*; Springer: Berlin, Germany, 2019.
9. Thangavel, M.; Varalakshmi, P.; Murrali, M.; Nithya, K. An Enhanced and Secured RSA Key Generation Scheme (ESRKGS). *J. Inf. Secur. Appl.* **2015**, *20*, 3–10. [CrossRef]
10. Tehranipoor, M.; Wang, C. *Introduction to Hardware Security and Trust*; Springer Science & Business Media: Berlin, Germany, 2011.
11. Zennaro, F.; Neri, E.; Nappi, F.; Grosso, D.; Triunfo, R.; Cabras, F.; Frexia, F.; Norbedo, S.; Guastalla, P.; Gregori, M. Real-Time Tele-Mentored Low Cost "Point-of-Care US" in the Hands of Paediatricians in the Emergency Department: Diagnostic Accuracy Compared to Expert Radiologists. *PLoS ONE* **2016**, *11*, e0164539. [CrossRef]
12. McCafferty, J.; Forsyth, J.M. *Point of Care Ultrasound Made Easy*; CRC Press: Boca Raton, FL, USA, 2020.
13. Daemen, J.; Rijmen, V. *The Design of Rijndael: AES-the Advanced Encryption Standard*; Springer Science & Business Media: Berlin, Germany, 2013.
14. Delfs, H.; Knebl, H.; Knebl, H. *Introduction to Cryptography*; Springer: Berlin, Germany, 2002; Volume 2.
15. Verbauwhede, I.M. *Secure Integrated Circuits and Systems*; Springer: Berlin, Germany, 2010.
16. Tuyls, P. *Towards Hardware-Intrinsic Security: Foundations and Practice*; Springer Science & Business Media: Berlin, Germany, 2010.
17. Shujun, L.; Xuanqin, M.; Yuanlong, C. Pseudo-random bit generator based on couple chaotic systems and its applications in stream-cipher cryptography. In Proceedings of the International Conference on Cryptology in India, Chennai, India, 16–20 December 2001; Springer: Chennai, India, 2001; pp. 316–329.
18. Wang, Y.; Wei, M.-D.; Negra, R. Low Power, 11.8 Gbps 2 7-1 Pseudo-Random Bit Sequence Generator in 65 nm standard CMOS. In Proceedings of the 2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Genova, Italy, 27–29 November 2019; IEEE: Genova, Italy, 2019; pp. 318–321.
19. Moon, T.; Tzou, N.; Wang, X.; Choi, H.; Chatterjee, A. Low-cost high-speed pseudo-random bit sequence characterization using nonuniform periodic sampling in the presence of noise. In Proceedings of the 2012 IEEE 30th VLSI Test Symposium (VTS), Maui, HI, USA, 23–26 April 2012; IEEE: Maui, HI, USA, 2019; pp. 146–151.
20. Bakiri, M.; Couchot, J.; Guyeux, C. CIPRNG: A VLSI Family of Chaotic Iterations Post-Processings for F_2 -Linear Pseudorandom Number Generation Based on Zynq MPSoC. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2018**, *65*, 1628–1641. [CrossRef]
21. Bakiri, M.; Couchot, J.; Guyeux, C. One random jump and one permutation: Sufficient conditions to chaotic, statistically faultless, and large throughput PRNG for FPGA. In Proceedings of the 14th International Joint Conference on e-Business and Telecommunications—SECRYPT, Madrid, Spain, 24–26 July 2017; Volume 6, pp. 295–302.
22. Ziller, A.; Usynin, D.; Braren, R.; Makowski, M.; Rueckert, D.; Kaissis, G. Medical imaging deep learning with differential privacy. *Sci. Rep.* **2021**, *11*, 13524. [CrossRef]
23. Klang, E. Deep learning and medical imaging. *J. Thorac. Dis.* **2018**, *10*, 1325–1328. [CrossRef]
24. Khan, M.F.; Saleem, K.; Alshara, M.A.; Bashir, S. Multilevel information fusion for cryptographic substitution box construction based on inevitable random noise in medical imaging. *Sci. Rep.* **2021**, *11*, 14282. [CrossRef] [PubMed]
25. Chhabra, S.; Lata, K. Obfuscated AES cryptosystem for secure medical imaging systems in IoMT edge devices. *Health Technol.* **2022**, *12*, 971–986. [CrossRef]
26. Joshi, S.; Mohanty, S.P.; Kougianos, E. Everything you wanted to know about PUFs. *IEEE Potentials* **2017**, *36*, 38–46. [CrossRef]
27. Saxena, N.; Voris, J. Data remanence effects on memory-based entropy collection for RFID systems. *Int. J. Inf. Secur.* **2011**, *10*, 213–222. [CrossRef]
28. Delvaux, J.; Verbauwhede, I. Fault injection modeling attacks on 65 nm arbiter and RO sum PUFs via environmental changes. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2014**, *61*, 1701–1713. [CrossRef]
29. Ghosh, S. Spintronics and security: Prospects, vulnerabilities, attack models, and preventions. *Proc. IEEE* **2016**, *104*, 1864–1893. [CrossRef]
30. Habibzadeh, H.; Nussbaum, B.H.; Anjomshoa, F.; Kantarci, B.; Soyata, T. A survey on cybersecurity, data privacy, and policy issues in cyber-physical system deployments in smart cities. *Sustain. Cities Soc.* **2019**, *50*, 101660. [CrossRef]

31. Najafi, F.; Kaveh, M.; Martín, D.; Reza Mosavi, M. Deep PUF: A Highly Reliable DRAM PUF-Based Authentication for IoT Networks Using Deep Convolutional Neural Networks. *Sensors* **2021**, *21*, 2009. [CrossRef]
32. Taneja, S.; Rajanna, V.K.; Alioto, M. In-Memory Unified TRNG and Multi-Bit PUF for Ubiquitous Hardware Security. *IEEE J. Solid-State Circuits* **2022**, *57*, 153–166. [CrossRef]
33. Gao, B.; Lin, B.; Li, X.; Tang, J.; Qian, H.; Wu, H. A Unified PUF and TRNG Design Based on 40-nm RRAM with High Entropy and Robustness for IoT Security. *IEEE Trans. Electron Devices* **2022**, *69*, 536–542. [CrossRef]
34. Rai, V.K.; Tripathy, S.; Mathew, J. Design and Analysis of Reconfigurable Cryptographic Primitives: TRNG and PUF. *J. Hardw. Syst. Secur.* **2021**, *5*, 247–259. [CrossRef]
35. Lotfy, A.; Kaveh, M.; Martín, D.; Mosavi, M.R. An Efficient Design of Anderson PUF by Utilization of the Xilinx Primitives in the SLICEM. *IEEE Access* **2021**, *9*, 23025–23034. [CrossRef]
36. Dameff, C.J.; Selzer, J.A.; Fisher, J.; Killeen, J.P.; Tully, J.L. Clinical cybersecurity training through novel high-fidelity simulations. *J. Emerg. Med.* **2019**, *56*, 233–238. [CrossRef]
37. Halak, B. *Hardware Supply Chain Security: Threat Modelling, Emerging Attacks and Countermeasures*; Springer Nature: Berlin, Germany, 2021.
38. Chang, C.-H.; Potkonjak, M. *Secure System Design and Trustable Computing*; Springer: Berlin, Germany, 2016.
39. Stinson, D.R. *Cryptography: Theory and Practice*; Chapman and Hall: Boca Raton, FL, USA, 2005.
40. Mishra, P.; Bhunia, S.; Tehranipoor, M. *Hardware IP Security and Trust*; Springer: Berlin, Germany, 2017.
41. Mukhopadhyay, D.; Chakraborty, R.S. *Hardware Security: Design, Threats, and Safeguards*; CRC Press: Boca Raton, FL, USA, 2014.
42. Turan, M.S.; Barker, E.; Kelsey, J.; McKay, K.A.; Baish, M.L.; Boyle, M. Recommendation for the entropy sources used for random bit generation. *NIST Spec. Publ.* **2018**, *800*, 102.
43. GetTickCount Function (sysinfoapi.h). Available online: https://learn.microsoft.com/en-us/windows/win32/api/sysinfoapi/nf-sysinfoapi-gettickcount (accessed on 8 October 2022).
44. GetSystemTime Function (sysinfoapi.h). Available online: https://learn.microsoft.com/en-us/windows/win32/api/sysinfoapi/nf-sysinfoapi-getsystemtime (accessed on 8 October 2022).
45. QueryPerformanceCounter Function (profileapi.h). Available online: https://learn.microsoft.com/en-us/windows/win32/api/profileapi/nf-profileapi-queryperformancecounter (accessed on 8 October 2022).
46. Tuyls, P.; Škoric, B.; Kevenaar, T. *Security with Noisy Data: On Private Biometrics, Secure Key Storage and Anti-Counterfeiting*; Springer Science & Business Media: Berlin, Germany, 2007.
47. Sadeghi, A.-R.; Naccache, D. *Towards Hardware-Intrinsic Security*; Springer: Berlin, Germany, 2010.
48. Hough, D. Applications of the proposed IEEE 754 standard for floating-point arithetic. *Computer* **1981**, *14*, 70–74. [CrossRef]
49. Katz, J.; Menezes, A.J.; Van Oorschot, P.C.; Vanstone, S.A. *Handbook of Applied Cryptography*; CRC Press: Boca Raton, FL, USA, 1996.
50. Stallings, W. *Cryptography and Network Security: Principles and Practice*; Pearson Education: London, UK, 2003.
51. Blake, I.; Seroussi, G.; Seroussi, G.; Smart, N. *Elliptic Curves in Cryptography*; Cambridge University Press: Cambridge, UK, 1999; Volume 265.