Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 3173995, 8 pages
https://doi.org/10.1155/2022/3173995

Research Article

A Hash-Based Fast Image Encryption Algorithm

Ruifeng Han

Computer Science Department, Xinzhou Teachers University, Xinzhou 034000, China

Correspondence should be addressed to Ruifeng Han; hrf_xztu@163.com

Received 1 March 2022; Accepted 13 July 2022; Published 10 August 2022

Academic Editor: Ruinian Li

Many specialists and academics have recently become interested in the security of digital images in applications for the Internet of Things. Hash-based digital image encryption algorithms with high unified average changing intensity (UACI > 30.96 percent) and only one pixel difference from the plain image would therefore adjust plenty of the pixels in the cipher image and have indeed been suggested to maintain the protection of images in the Internet of Things (NPCR > 98.77 percent). Theoretical study and simulation results show that the suggested approach can fix these issues while retaining all the advantages of the original. The proposed image encryption algorithm has important application value for strengthening the security of the Internet of Things.

1. Introduction

With the continuous development of information technology and computer processing power, cryptography has also been extended. Because the network environment, especially Internet of Things environment is vulnerable to attacks, and digital images contain redundant information, which is closely related to personal privacy, once it is leaked, it is likely to be personally threatened and even affect commercial secrets and national security. Many algorithms exclusively handle with digital encrypted images or text encryption, respectively. Several algorithms, nevertheless, integrate these two tasks. Passwords are simpler and easier for people to recognize than pseudorandom numbers or a long string of codes, making it more practical and appropriate to password-protect digital photographs [1]. Liu and Tan put forth a plan that uses 1D SHA-2 algorithms combined with password protection to encrypt digital images with compound forward transform [2].

2. Defect Analysis of the Original Algorithm

2.1. Lack of Connectivity between Pixels during Encryption. During the encryption of the original image, the key stream generation step and there really is no relationship either between pixel intensity, and indeed, the postprocessing phase solely affects each pixel inside a one-to-one relationship. This characteristic may expose it to selected attacks [3]. An attacker can do this by encrypting only two images that differ by only one pixel. The plain picture and the cipher image could both be found by an opponent because the two encrypted images just vary with one pixel from one another. Figure 1 illustrates an illustration of a 16×16 image. Following encrypted data, point (1, 2) in the original image is transferred to point (12, 5). An opponent can find the permutation rules for a pixel by performing this method again for each pixel in the original image. This is the same as disclosing a fresh random map during postprocessing [4].

2.2. The Only Dependency of the Key Stream. We discover that the participant's password is completely dependent on the secret key during the encryption step of the original technique. The associated stream usually stays the same as long as the cipher is unmodified, regardless of whether this is used mostly for XOR with the original picture or for a postprocessing step. A known-plaintext assault on the original algorithm is hence inevitable. The following formulas are used to express the random key $\{K1, K2, \cdots, KM \times N\}$, KMN that the adversary could retrieve if they are given a pair of pure image needs to set $G = \{G1, G2, \cdots, GM \times N\}$ and a placed of cipher maps $C = \{C1, C2, \cdots, CM \times N\}$.

$$K_i = C_i \oplus G_i, \quad (1 < i \leq 8 \times M \times N). \tag{1}$$

The permutation rules for pixels are recovered when

Figure 1: Mapping process.

used with the chosen-plaintext attack previously discussed. Understanding the key stream produced by a specific key is identical to actually having that key. The equivalent normal image may be retrieved instantly for whichever cipher image encoded with almost the same key.

2.3. Attack Simulation. Let us say there is a password image but no one knows the key. An attacker could trick the encryption system by feeding it a black image (every pixel inside the encrypted image equals 0) to get the cipher image *B*. *B* is definitely a different key stream. We could derive the replacement rule, designated by *M*, using the procedure outlined in Section 2.1. Apply rule *M* on the replaced key stream to undo it. We could create a new key stream that seems to be identical to the key stream obtained by doing an XOR operation with the original image prior to replacement. There seem to be two procedures required to obtain the original image for encrypted images. Flip the encrypted image using rule M firstly and afterwards XOR the new key stream after that. The outcome is the accurate normal picture. Throughout Figure 2, the attack procedure is depicted.

3. Improved Algorithm

In order to overcome the above potential defects, the following improved algorithm is proposed:

(1) For convenience, the initial image post-processing step is modified by us to the following equation, defined as ACM; this includes the three variables *a*, *b*, and *k*. To create the three variables that will substitute the efficient algorithm, we utilize $Ki = H(H($ Key). In our enhanced algorithm, however, *k* rounds of operations are required in the encryption/decryption stage. On each turn, swap positions between (0,0) and (1,1). Each round takes turns.

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} ab+1 & a \\ b & 1 \end{pmatrix} \begin{pmatrix} x_n \\ y_n \end{pmatrix} (\text{mod } 1), \quad a, b \in N \tag{2}$$

(2) The improved algorithm is performed by splitting the image into two equal pieces and then independently encrypting each piece. The key stream is determined by the user key and original image features [5, 6]. In other words, when the same key is used, different images are encrypted with the key stream as a variable; it aids in preventing selected and known-plaintext attacks on the encryption algorithm

3.1. Encryption

(1) Identical to steps one and two of the initial step

(2) Generate a new key stream K_0 with image features. First, the original image is divided into two equal parts L_0 and R_0 in the vertical direction, and the two parts are encrypted, respectively, and the formula is as follows. First, encrypt the information of R_0 using L_0 to R_1; the right part has been encrypted. Then, L_0 is encrypted to L_1 using R1's information

(3) Finally, to acquire the cipher picture, combine the 2 encrypted components L_1 and R_1

$$\begin{cases} R_1 = \text{Complex}\left(\text{imresize}\left(\left(K' \oplus g_{L_0}\right), \left[M, \frac{N}{2}\right], '\text{nearest}'\right)\right) \oplus R_0, \\ L_1 = \text{Complex}\left(\text{imresize}\left(\left(K' \oplus g_{R_1}\right), \left[M, \frac{N}{2}\right], '\text{nearest}'\right)\right) \oplus L_0. \end{cases} \tag{3}$$

L_0 and R_0 throughout the example above stand for the left and right sides of something like the pure picture, while L_1 and R_1 stand for the left and right sides of the cipher image, g_{L_0} and g_{R_1} represent the average gray value of L_0 and R_1 which is of variable size, (•, $[M, N/2]$, "nearest") represents the expansion operation, the "nearest" method is used to the process of enlarging an image to *M* rows, and $N/2$ column is known as cosine similarity interpolation. The number of the pixel to which this pertinent is allocated to the output pixel; additional pixels are not taken into account. The compound transformation in the initial encryption is represented by complex(•)

(4) The postprocessing step is ACM

(5) Encrypted images

Figure 3 depicts the encryption process algorithm. Figure 4 depicts the key stream creation algorithm.

3.2. Decryption. To obtain the XOR key stream, firstly invert the ACM.

Encrypt the XOR key stream to produce a plain picture next. L_1 to L_0 first were encrypted using R_1's data. Next, as stated in the following expression, utilize the data from L_0 to decode R_1 to R_0. L_0 and R_0 are the left and right halves of the plain image, whereas L_1 and R_1 are the left and right halves of the encrypted image.

$$\begin{cases} L_0 = L_1 \oplus \text{Complex}\left(\text{imresize}\left(\left(K' \oplus g_{R_1}\right), \left[M, \frac{N}{2}\right], '\text{nearest}'\right)\right), \\ R_0 = R_1 \oplus \text{Complex}\left(\text{imresize}\left(\left(K' \oplus g_{L_0}\right), \left[M, \frac{N}{2}\right], '\text{nearest}'\right)\right). \end{cases} \tag{4}$$

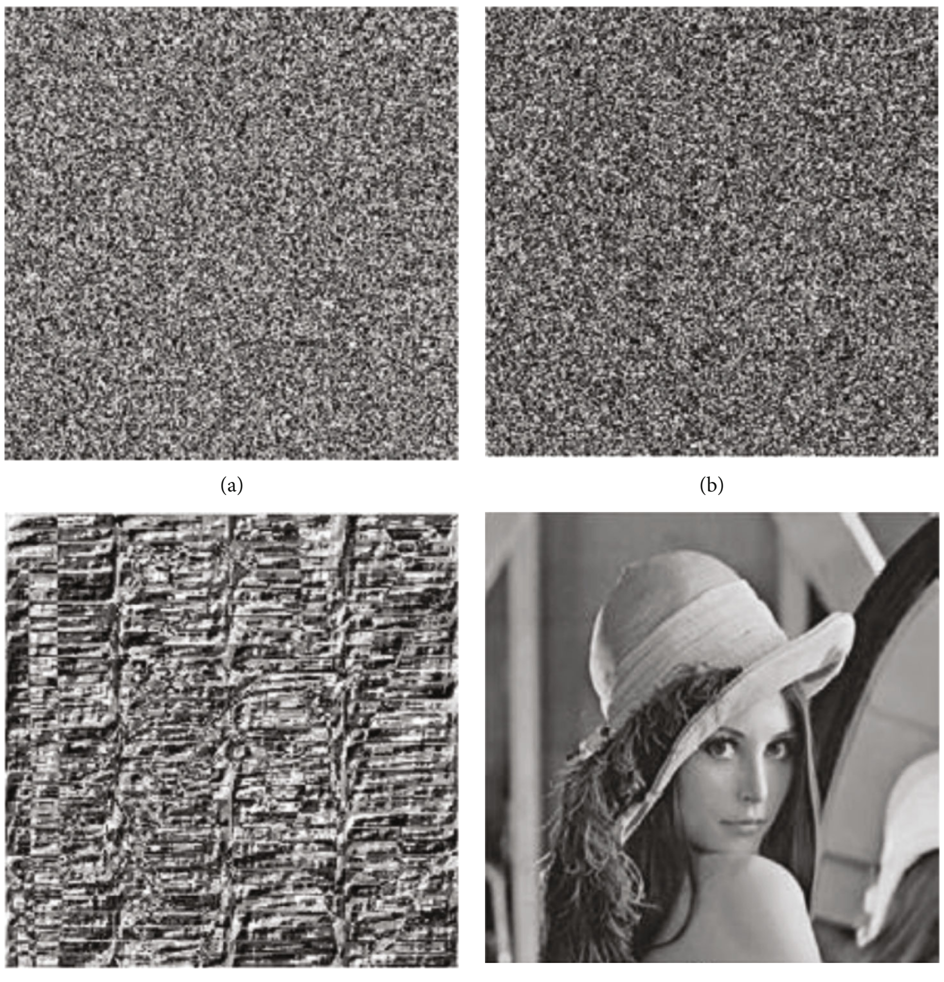

(a) (b) (c) (d)

Figure 2: Attack process.

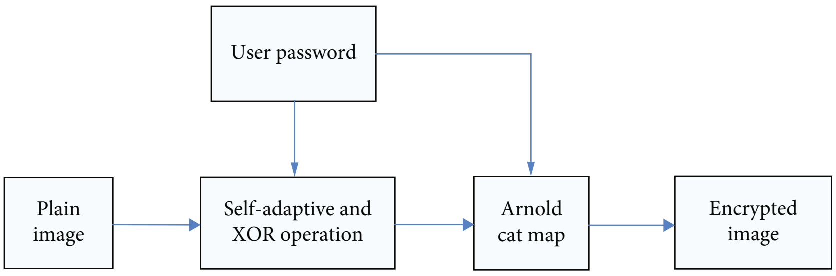


Figure 3: Encryption process algorithm.

Finally, the two decrypted parts L_0 and R_0 are connected together to obtain a decrypted image.

4. Performance Analysis

In this section, a 256×256 Lena gray image [7] is selected to conduct comparative experiments.

All experiments were performed using MATLAB 7.9 on a personal computer (PC) which has a 250 GB hard drive, a 2.0 GHz Intel dual-core microprocessor, and 1.99 GB of storage [8].

4.1. Histogram of Encrypted Image. Every gray level's frequency is shown in the graph, and that every gray level mainly related to the digital image [9]. The histograms of the original and encrypted images are seen in Figure 5.

The graph of the password picture is quite homogeneous and distinct from the actual picture, as shown in the figure.

4.2. Correlation of Two Adjacent Pixels. The correlation of test image pixels includes horizontal correlation and diagonal correlation: first, in the picture, 2500 pairs of adjacent

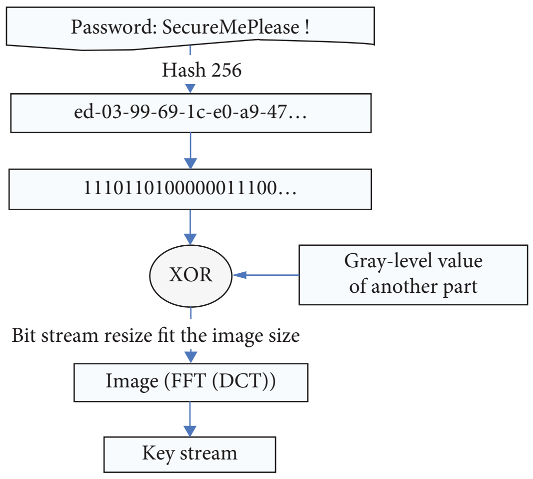
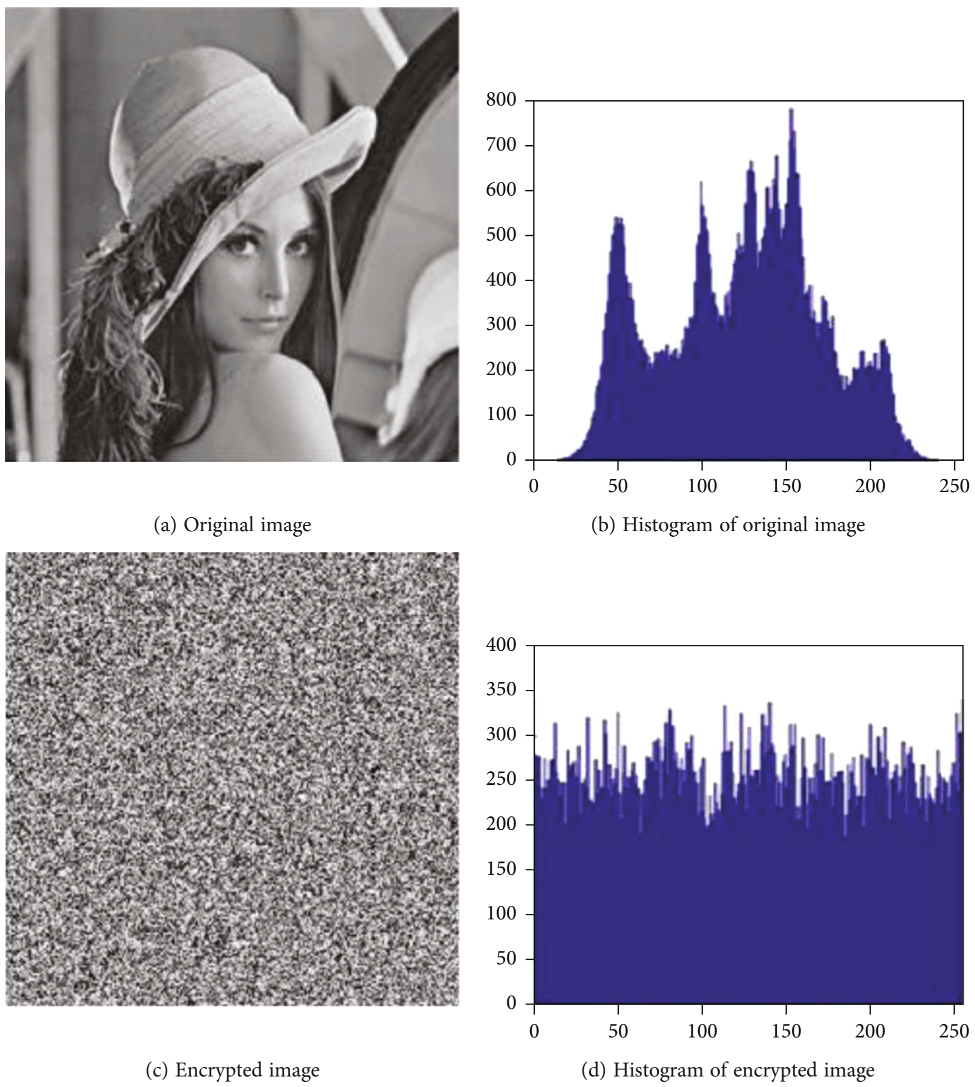

Password: SecureMePlease !
Hash 256
ed-03-99-69-1c-e0-a9-47...
1110110100000011100...
XOR
Gray-level value of another part
Bit stream resize fit the image size
Image (FFT (DCT))
Key stream

Figure 4: Key stream generation algorithm.



(a) Original image (b) Histogram of original image

(c) Encrypted image (d) Histogram of encrypted image

Figure 5: Histogram of original image and password image.

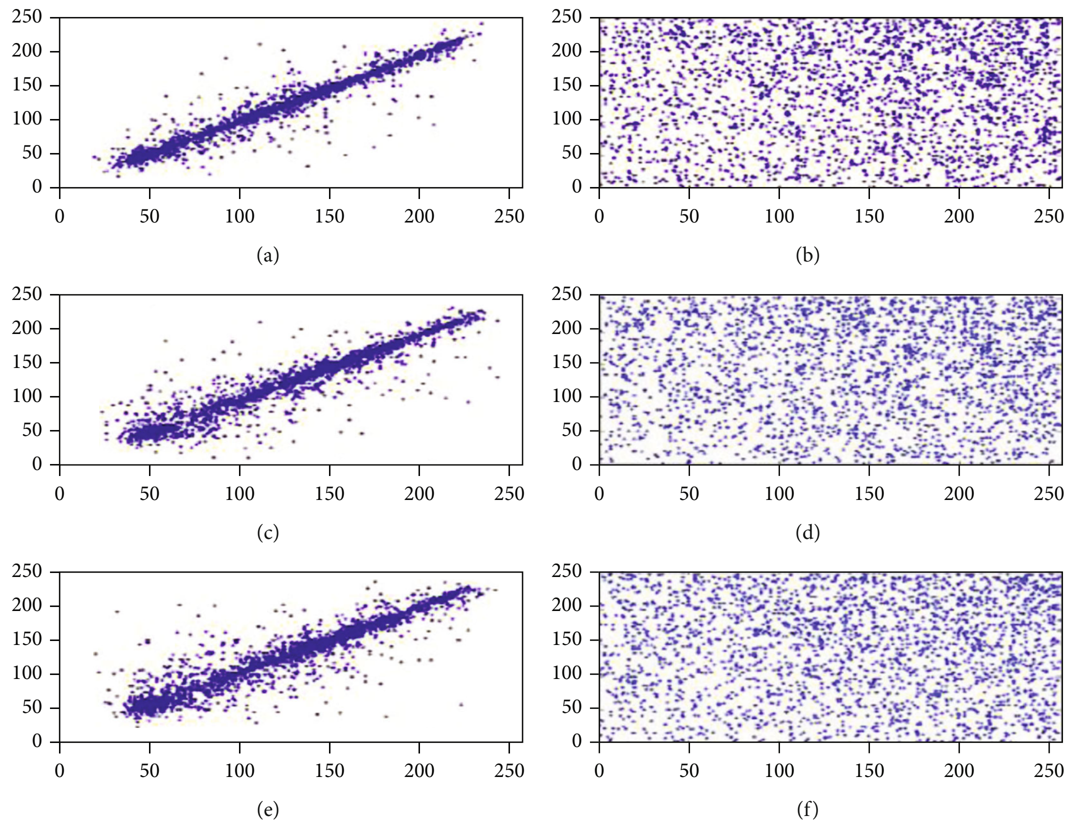


Figure 6: Correlation of two horizontally adjacent pixels.

Table 1: Correlation coefficient of two adjacent pixels.

	Normal image	Password image
Level	0.9431	0.0089
Vertical	0.9725	-0.0215
Diagonal	0.9264	-0.0074

Table 2: NPCR and UACI analyses.

	Normal image	Password image
NPCR	0.0015%	98.7778%
UACI	0.0000%	30.9639%

pixels are chosen at random, and then, the correlation coefficient is calculated by

$$r_{xy} = \frac{|\mathrm{cov}\,(x, y)|}{\sqrt{D(x)} \times \sqrt{D(y)}}, \tag{5}$$

$$\mathrm{cov}\,(x, y) = \frac{1}{N}\sum_{i=1}^{N} x_i, \tag{6}$$

$$D(x) = \frac{1}{N}\sum_{i=1}^{N} (x_i - E(x))^2, \tag{7}$$

where the gray scale values of the adjacent image pixels are represented by x and y.

Figure 6 depicts, correspondingly, the horizontal, vertical, and diagonal correlations between the two images. Table 1 displays the findings of the regression analysis of adjacent pixels.

4.3. Sensitivity Analysis. Often, an adversary might alter the encrypted image slightly in order to observe the change in results. In doing so, meaningful relationships between plain images and cryptographic images can be found. This is called a differential attack.

4.3.1. NPCR and UACI Analyses. But only when the pictures vary by something like a single pixel, NPCR denotes the rate of difference in the frequency of pixels inside an encryption algorithm. The unified average intensity of change (UACI) metric calculates the average brightness of the variation between the two images. NPCR and UACI both rely on slight adjustments to the two images while maintaining the same key. In reference [10], the original image and key of the NPCR calculated by the author have changed by one bit.

Suppose there are two encrypted pictures, C_1 and C_2, which normal control pictures vary by just one pixel. The gray scale values of the encrypted images C_1 and C_2 are designated as $C_1(i, j)$ and $C_2(i, j)$, accordingly, in the i-th row and j-th column.

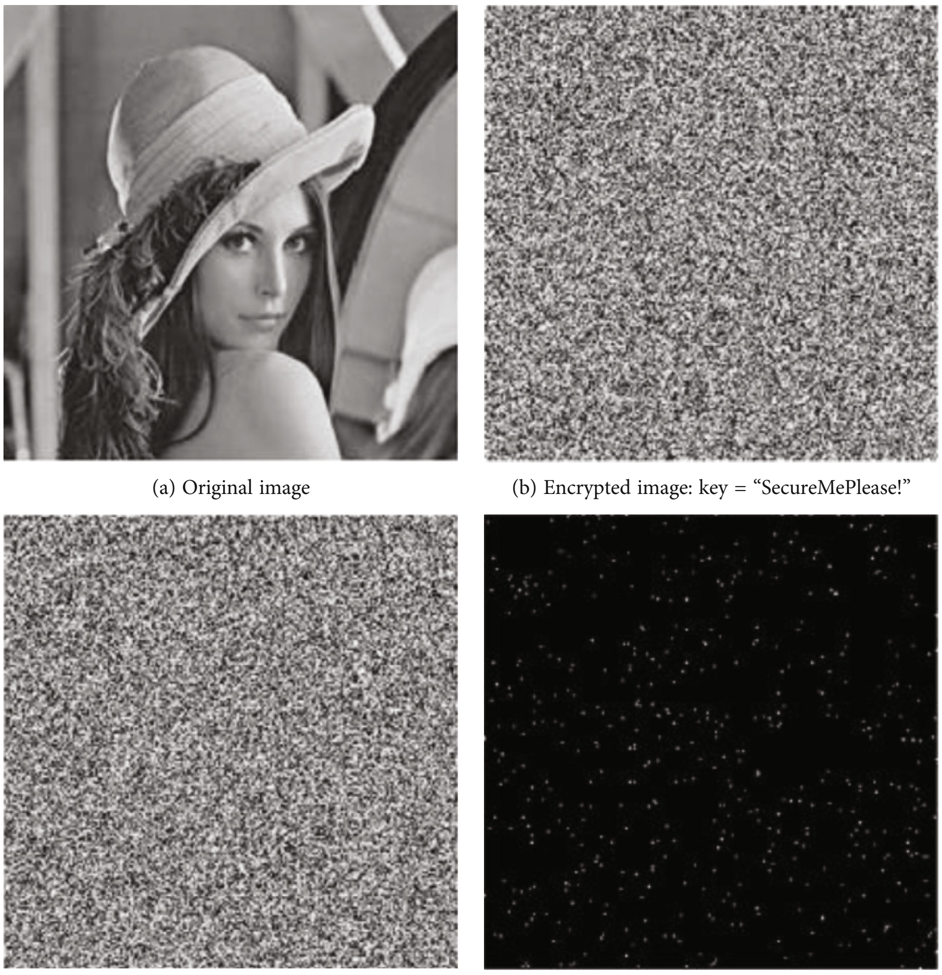

(a) Original image (b) Encrypted image: key = "SecureMePlease!"
(c) Encrypted image: key = "SecureMePlease!" (d) Difference image

Figure 7: Key sensitivity test.

Table 3: Rate of change.

Original key	New key	Change rate
SecureMePlease!	SecureMePlease.	99.600%
	secureMePlease!	99.161%
	SecuremePlease!	99.245%
	SecureMePlease	99.295%

The definitions of NPCR and UACI are

$$\mathrm{NPCR} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}}{M \times N} \times 100\%,$$
$$\mathrm{UACI} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}\left|C_i(i,j) - C_2(i,j)\right|}{M \times N \times 256} \times 100\%. \quad (8)$$

The sizes of the picture are M and N, correspondingly.

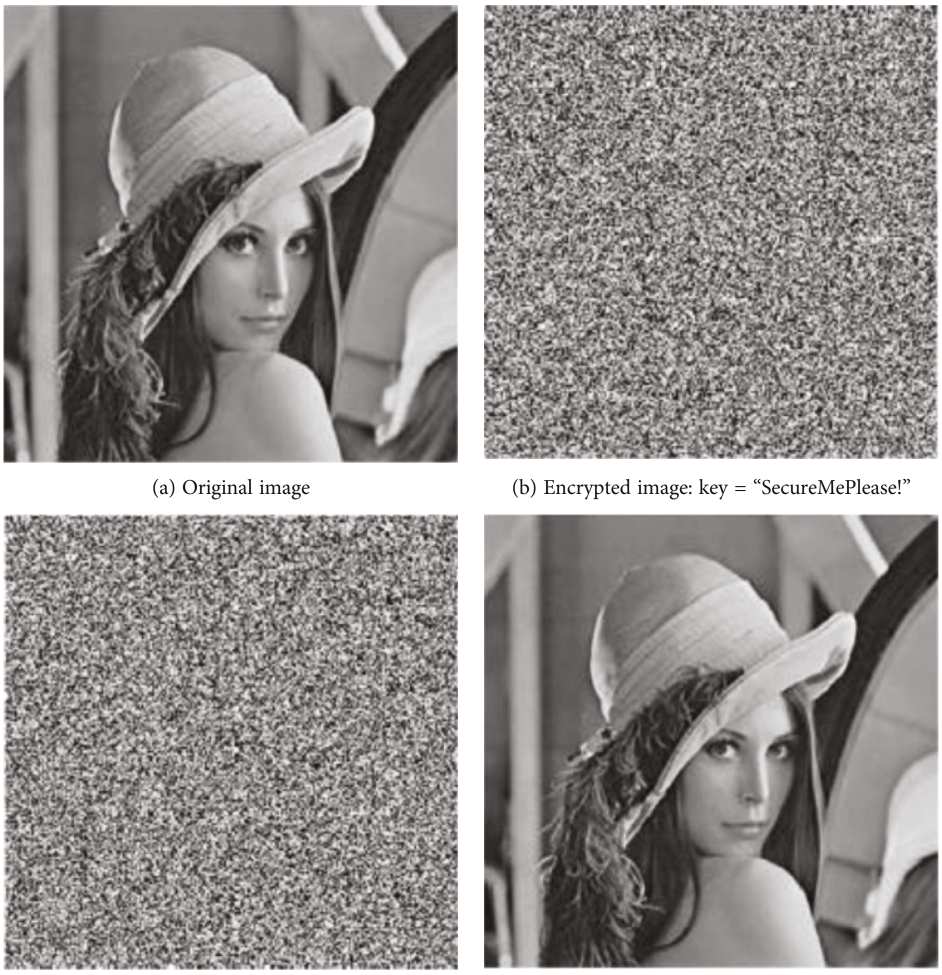
For our instance, the key "SecureMePlease!" has been used, and the chosen pixel is the position (1,2) of the image pixels. Its number is altered from (10100011)2 to (10100010)2. NPCR and UACI are therefore reported in Table 2.

Table 2 demonstrates how much greater our revised technique is programmatically than the original method in sensitivity.

The responsiveness to the actual picture is quite low in the original method. The whole key picture in the 256×256 sample provided has only been modified by roughly 0.0015 percent. The UACI is around 0 [11]. Experiments show that encrypted images are vulnerable to plaintext and known-plaintext attacks.

Consequently, it is evident that our enhanced approach fixes the old system's flaw of being blind to minute alterations in planar images. The encrypted picture's pixels shift whenever a single pixel in the original picture changes ($\mathrm{NPCR} > 98.77\%$), and the universal mean shift intensity increases ($\mathrm{UACI} > 30.96\%$). The results demonstrate that the effectiveness is satisfactory.

4.3.2. Key Sensitivity Test. Crucial sensitivity is the pace at which a cryptographic picture's pixel count changes even though only one piece of the password is changed. The usual image should first be encrypted with the testing password "SecureMePlease!" before the least-valid password is changed to "SecureMePlease." and the identical image is encrypted. Consider the various password photos once more.



(a) Original image

(b) Encrypted image: key = “SecureMePlease!”

(c) Decrypted image: key = “SecureMePlease!”

(d) Decrypted image: key = “SecureMePlease!”

Figure 8: Key sensitivity test.

Table 4: Encryption time of three encryption algorithms.

Encryption algorithm	Image size	Encryption time
Literature [12]	90*180	2.13
Literature [13]	128*128	1.24
The algorithm in this paper	128*128	0.15

The outcome is in regard to pixel gray levels, the encrypted image is completely different with a slight difference in the key (as shown in Figure 7). Table 3 displays the outcomes of the encryption with different keys, with an average rate of change as high as 99.323%.

Also, if the image is encrypted with a key and decrypted with another, simply modified key, the decryption fails. Figure 8 demonstrates how photos encrypted using the “SecureMePlease!” password cannot be properly decoded with the “SecureMePlease” password.

4.4. Other Security Analysis

4.4.1. Resist Plaintext Attack. We may observe a significant change at the crucial stream creation when evaluating the original method with the revised approach. The user’s passcode is the single factor that determines how the key stream is utilized in the original file, whether it is for an XOR operation with the actual picture or for postprocessing. It is separated from typical visuals. The key stream in our enhanced technique is based on the original picture’s properties as well as the user’s passcode. That is, although just use the same passcode, if various photos are encoded, the crucial stream of the XOR stage is unpredictable. This person assists with encrypted photographs defend against selected plaintext attacks and known plaintext attacks.

Also, a cryptanalyst sending a dark image through into encryption method seems to have no impact on the operation because the key stream is varied when encrypting various images with the same cipher. The “chosen plain text assault” discussed above can be eliminated by our enhanced algorithm.

4.4.2. Diffusion and Chaos. Obfuscation and diffusion are two characteristics of secure cryptographic procedures that were first established by Claude Shannon in the field of cryptography. NPCR demonstrates that when just one pixel of the plaintext is altered, nearly every pixel in the cryptographic picture is altered, as illustrated in Section 4.3.1 (NPCR > 98.77 percent). The revised algorithm's diffusion characteristics are excellent.

Through the correlation analysis of the gray histogram and adjacent pixels, the proposed improved algorithm has good chaos.

4.4.3. Brute Force Attack. The suggested enhanced algorithm depends on the required space length and critical sensitivities for brute force attack evaluation.

Essential space: the SHA-2 method and the FFT-DCT composite transform are irreparable processes, as was already stated; in addition, ARM will arrange the pixels, which is very secure for common commercial applications

Key sensitivity: as shown in Figures 7 and 8, even a small key change causes almost all pixels to change the corresponding cryptographic image

Therefore, our algorithm is highly resistant to brute force attacks.

4.5. Comparison of Similar Algorithms. The algorithm proposed in this paper has superiority by comparing and analyzing the algorithm of literature [12] and literature [13]. Reference [12] suggested a Feistel network-based picture encryption technique, which has high security and can be comparable to the algorithm in this paper, but there is a gap with this paper in terms of sensitivity. This paper also introduces in Section 4.3. The algorithm in this paper Fewer iteration rounds are required.

In addition, although the literature [13] has been improved, the encryption speed of the algorithm in this paper is significantly higher than that of the literature [13]. Table 4 shows the image encryption algorithms proposed in this paper, literature [12] and literature [13]. The time required to encrypt the same image, it can be seen that the encryption speed of this paper is the fastest that is because the literature [12] uses a large number of iterations, resulting in a slow operation, while the literature [13] is because there are too many rounds. This results in increased computation time, which in turn slows down encryption.

5. Conclusion

In this research, we suggested a hash-based fast picture encryption algorithm for Internet of Things (IoT) applications, where the image is split into equivalent left and right portions, and the data from one half is used to encrypt the other part in turn. Theoretical study and computer simulation demonstrate the robustness of our suggested approach against chosen plaintext assaults as well as chosen plaintext attacks.

Data Availability

All the data used to support the findings of this study are available in the article.

Conflicts of Interest

No contradictions exist, according to the researchers, with the publication of this research.

References

[1] H. Dai, W. Dong, and S. Zhong, "Design and analysis of a class of SHA-x improved hash algorithms," *Computer Engineering*, vol. 35, no. 6, pp. 181-182+185, 2009.

[2] R. Liu and T. Tan, "A review of research on digital image watermarking," *Journal of Communications*, vol. 21, no. 8, pp. 40–49, 2000.

[3] J. Wu and S. Song, "An improvement of text encryption method," *Journal of Chongqing University of Science and Technology: Natural Science Edition*, vol. 6, no. 2, pp. 55-56, 2004.

[4] A. Zhou and Y. Yu, "Robust speech recognition by adopting random projection in feature space," *Computer Applications*, vol. 32, no. 7, pp. 2070–2073, 2012.

[5] X. Liao, S. Lai, and Q. Zhou, "A novel image encryption algorithm based on self-adaptive wave transmission," *Signal Processing*, vol. 90, no. 9, pp. 2714–2722, 2010.

[6] D. Xiao and F. Y. Shih, "Using the self-synchronizing method to improve security of the multi chaotic systems-based image encryption," *Optics Communications*, vol. 283, no. 15, pp. 3030–3036, 2010.

[7] R. Liu, F. Li, and L. Su, "Bilateral filtering based image restoration for multiple grayscale images," *Computer Applications*, vol. 30, no. 4, pp. 902–904, 2010.

[8] Y. Li, "Simulation calculation of control system—MATLAB," *Computer Measurement and Control*, vol. 12, no. 4, pp. 40–43, 1996.

[9] Y. Tang, Z. Wwang, and J. A. Fang, "Image encryption using chaotic coupled map lattices with time-varying delays," *Communications in Nonlinear Science and Numerical Simulation*, vol. 15, no. 9, pp. 2456–2468, 2010.

[10] S. Deng, Y. Zhan, and D. Xiao, "Analysis and improvement of a hash-based image encryption algorithm," *Communications in Nonlinear Science & Numerical Simulation*, vol. 16, no. 8, pp. 3269–3278.

[11] A. Manikond and P. Mangalampalli, *UACI: Uncertain Associative Classifier for Object Class Identification in Images*, 2012.

[12] F. Li and J. Xu, "Image encryption algorithm based on hash function and multi-chaotic system," *Computer Engineering and Design*, vol. 31, no. 1, pp. 141–144, 2010.

[13] G. Chen, X. Zhao, and J. Li, "A self-adaptive algorithm on image encryption," *Journal of Software*, vol. 16, no. 11, p. 1975, 2005.