



Hybrid Image Encryption Scheme Using SPECK Block Cipher and Salsa20-Based DNA-Enhanced Key Generation

Amjed Abbas Ahmed^{1,2}, Zainab Rustum Mohsin³, Hayder Najm², Taher M. Ghazal^{2,4}, Mohammad Ahmed Alomari⁵, Adam Wong Yoon⁶

¹Center for Cyber Security, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Bangi, Malaysia

²Department of Computer Techniques Engineering, Imam Alkadhim University College, Baghdad, Iraq

³College of Computer Science and Mathematics, University of Thi-Qar, Thi-Qar, Iraq

⁴Department of Networks and Cybersecurity, Hourani Center for Applied Scientific Research, Al-Ahliyya Amman University, Amman, Jordan

⁵Department of Cybersecurity, Faculty of Artificial Intelligence and Cyber Security (FAIX), Universiti Teknikal Malaysia Melaka (UTeM), Melaka, Malaysia

⁶Department of Engineering Technology, Fakulti Teknologi Dan Kejuruteraan Elektronik Dan Komputer (FTKEK), Universiti Teknikal Malaysia Melaka (UTeM), Melaka, Malaysia

E-mails: amjedabbasahmed@gmail.com taher.ghazal@ieee.org alomari@utem.edu.my
(corresponding author) adamwong@utem.edu.my

Abstract: This paper presents a new hybrid image encryption algorithm, which combines the SPECK block cipher with a Salsa20-based key generation process. Although these factors have been integrated in the past, what is unique about our work is that DNA encoding and transformation are applied only to the key stream, not to the image data itself. The method produces a non-linear, irreversible key obfuscation layer that enhances the system's resistance to key-based attacks at a relatively small cost to the overall computation of the main image encryption operation. The system will have the key generated by Salsa20 to 128 bits, converted by DNA complement and reverse operations, and finally loaded into the SPECK cipher to be encrypted. The approach provides a high-security level but with memory-efficient computation, and, therefore, it is resource-constrained and can be applied to IoT devices. The system shows robust results in NPCR (>99.6%), UACI (~33%), and entropy (7.99).

Keywords: Data security, Cybersecurity, Hybrid image encryption, Block cipher, DNA encoding.

1. Introduction

Lightweight cryptography refers to cryptographic primitives, techniques, and ciphers that are optimized for low-resource devices with limited memory, power, or processing capacity [1, 2]. The dilemma in this field is that three primary factors should be considered: security, performance, and cost. Although optimization of two aspects, such as security and performance, cost and efficiency, or security and cost, may often be possible, an optimal trade-off among all three is challenging. Consequently, a large number of lightweight cryptography systems have been established, embodying their applications in software, as well as in hardware [3-5].

Digital images bear sensitive personal and organizational information over the networks, e.g., medical, satellite, biometric scans, and document digitalization, among others, with distinctive qualities. Like normal data, such images face many security threats [6, 7]. Nevertheless, they have other specific characteristics, such as high redundancy, robust correlation of the pixels, and massive size. Also, various imaging applications have particular requirements, including the maintenance of visual quality, support of effective compression of the application to make transmission possible, or permit real-time processing [8]. The solutions offered by traditional encryption schemes, which were primarily written to support text, do not always consider these specific needs or even ignore adjustments using algorithm-specific requirements that require information-specific characteristics, such as algorithms to support images [9].

Image encryption refers to encoding a visual picture into a non-readable form (which usually appears like noise) such that only authorized groups can read and access the original content. Encryption and Decryption are the two primary operations involved in the process of cryptography, wherein a plain version of an image would be converted into a cipher image and vice versa [10]. Images are a crucial component in modern digital communication, and with the swift increase in IoT applications, high-speed multimedia transfers can be achieved. Nevertheless, it is hard to ensure sensitive visual information security because IoT devices are still limited in memory and computational capabilities. However, Cellular Automata (CA) potentially provide an alternative to IoT settings that is not typically available with traditional encryption methods. These are applicable in lightweight encryption with the resources of IoT devices since a CA can quickly produce complicated pseudo-random sequences. They are also simple to implement because they are hardware-friendly [11-14].

This paper presents a new image encryption scheme, a combination of the SPECK-128 block cipher and Salsa20-based key generation, and a new DNA-based key obfuscation layer. The main contribution of this work is two-fold:

1. First Use of DNA Operations on the Key Stream: In contrast to all the known DNA-based image encryption techniques [15, 18-20] that encode DNA information on image pixels (and therefore require $O(N^2)$ operations, our proposed technique relegates all DNA manipulations to the key stream (128 bits). The basic change reduces the computational complexity from $O(N^2)$ to $O(N)$ and yields similar security indicators (NPCR = 99.6, entropy = 7.99).

2. New Non-Linear Key Obfuscation Layer: This entails a sequence of DNA functions (encoding, complement, reverse, decoding) to form a non-linear transformation between the Salsa20 keystream and the SPECK master key. This layer removes any possible linear associations that can be used against it in key-recovery attacks, including cryptographic attacks, without altering the fixed SPECK cipher structure.

The remainder of this paper is structured as follows. Section 2 examines related literature in hybrid image encryption. Section 3 outlines the proposed system architecture, including the Salsa20-based DNA-enhanced key generation process, image preprocessing, and SPECK's encryption/decryption process. Section 4 provides and argues the experimental results, security measures, performance analysis, and a comparison with the existing works. Finally, Section 5 concludes the paper.

2. Related works

Images are vital in contemporary interactions between individuals and organizations and may contain delicate information that needs to be sent safely. The data integrity and confidentiality in transferring images to various destinations are ensured through many methods. Hereafter, a brief related work is presented. Ahmed et al. [15] proposed a new DNA-based color image encryption model that encodes large HD images using a convolutional auto-encoder to resolve the latency issue on the encrypted image. The new method, proposing the theory of chaos, DNA computing, and the usage of S-boxes, will provide higher levels of security. In contrast, the autoencoder is implemented to improve processing speed due to reduced dimensionality. The quality of encryption in the experiment is high, as the values of entropy C (7.99), NPCR (>99%), and UACI (>33) support the assertion that it is robust to attacks. The scheme exceeds current key space size and computational efficiencies (10^{285}). Muhalha and Alshawhi [16] proposed a new lightweight

hybrid encryption algorithm, the Speck-Salsa20 Algorithm for Data Integrity and Confidentiality (SSDIC), which is proposed to resolve such flaws as non-randomness and susceptibility to side-channel attacks in the Speck block cipher. The SSDIC is a lightweight hybrid of the Speck block cipher and the Salsa20 stream cipher. SSDIC adds security and passes all NIST statistical tests with better randomness by replacing the key generation algorithm of Speck with the Salsa20 hash function and by changing the round function. Experimental results indicate that SSDIC delivered 66% greater performance than standalone Speck in data integrity and confidentiality and was efficacious on resource-constrained IoT objects. The hybrid method utilizes the speed of Salsa20 and the flexibility of Speck, thus being adapted to low-power purposes. Naif et al. [17] proposed a 512-bit hybrid encryption algorithm, EAMSA 512, based on a modified Salsa20 stream cipher, chaotic systems, and S-boxes to increase security. Input data is divided into two 256-bit chains, one encrypted using chaotic Salsa20 and the other using S-boxes, swapped, and then 16 rounds are performed. Based on an 11-dimensional chaotic key generator, EAMSA 512 has strong randomness, with all NIST tests meeting the ideal score of nearly 7.99 (entropy) and 0.001 (correlation). Benchmarking is faster than AES, Speck, and GOST, in fact, by a significant margin, with encryption of 1KB of data taking only 0.097 ms. It is lightweight, IoT- and cloud-ready, and resistant to statistical and differential attacks. Mohammed, Khodher and Alabaichi [18] proposed a new lightweight image encryption algorithm integrating a 5D chaotic system, DNA computing, and Salsa20 to strengthen the performance and security. It uses a dynamic S-Box using Hn map information, and the diffusion is high with chaotic series information. Entropy (7.9965–7.9982), NPCR (0.99710), and UACI (33.68) are good performance parameters that resist statistical and differential attacks. In comparison tests, the correlation coefficients and execution time are lower, which proves that it applies to real-time applications.

Farooqui et al. [19] proposed a lightweight image encryption algorithm using DNA coding, snake permutation, and chaotic replacement to optimize security with a reduced computational overhead. The mechanism uses DNA coding to provide diffusion and chaotic S-boxes to provide confusion, which gives high randomness through its entropy of 7.9895 and low correlation of -0.001660 . The robustness against attacks is established by statistical tests of uniformity and contrast of histograms (10.1515). Specifically tailored to the limitations of IoT, the algorithm is efficient yet secure. Huang et al. [20] proposed a DNA-based image encryption algorithm that uses a 5D Hamiltonian conservative chaotic system and bit-plane decomposition to increase security and efficiency. Its process encodes

images onto DNA strands and conceals high-bit-plane data in code. A series of seven DNA operation index tables utilizes diffusion to attain high randomness (entropy 0.999) and attack resistance. It has an NPCR (99.61) performance and a UACI (33.46) performance that shows resistance to statistical and differential attacks. The algorithm is also resistant to cropping and a noise attack and has a low computational overhead.

3. Proposed system

The architectural philosophy of the proposed system lies in its difference from current hybrid encryption schemes. Whereas the previous literature encodes DNA operations into the image encryption process itself, interrupting each pixel with a series of biological transformations, we are using a key-centric model. In our system, as shown in Fig. 1, the DNA operations are limited to a preprocessing step that improves the cryptographic key before it is input to the SPECK cipher. This level of improvement is a paradigm shift: we do not increase the complexity of the image encryption process, but rather raise the resistance of the key to analysis, thereby securing the whole system at a basic level.

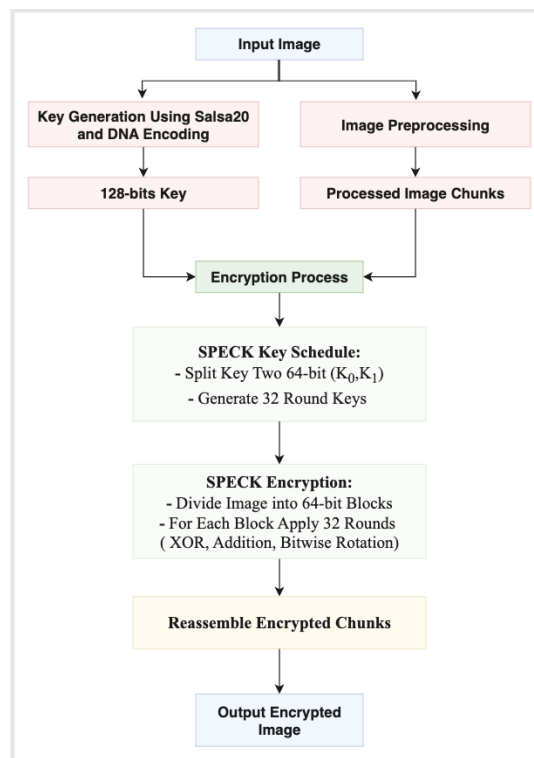


Fig. 1. Block diagram of the proposed system

3.1. Key generation using Salsa20 and DNA encoding

Whereas direct application of DNA operations to image data has been employed in related studies [15, 18-20], directly to the encryption key is a different strategy. The output of Salsa20 is pseudo-random already. However, the following DNA manipulations (complement and reverse) introduce non-linearity into the key material. This step generates no randomness; however, its security lies in disrupting any possible linear relationship between the Salsa20 output (keystream) and the master key used in SPECK. Although an attacker could examine the SPECK cipher and partially reverse the DNA transformations to recover the original Salsa20 output, without the precise encoding rules, it is non-trivial to reverse the DNA transformations and restore the original Salsa20 output, adding complexity to key recovery attacks.

As shown in Algorithm 1, the fundamental description of the key generation process is that of a two-step process developed to generate a cryptographically secure and obfuscated master key. The first phase utilizes the Salsa20 stream cipher to create a good pseudorandom seed. This seed is then subjected to a second phase of biological-inspired transformations to introduce a non-linear level of complexity and render the result, the key, more resistant to analysis.

The algorithm is initiated by a Salsa20 algorithm which uses a 256-bit secret key, a 64-bit nonce, and an 8-byte (64-bit) counter set at 0. To this end, 64 bytes of keystream are generated. This keystream is then peeled off the first 16 bytes (128 bits) to create what is termed the initial key material, which is defined to be the Salsa20 key. This is a typical and effective way of extracting a fixed-length key out of the output of a stream cipher.

The DNA obfuscation and encoding of the Salsa20 key occur after the Salsa20 generation. The key (128 bits) is then converted into a binary version of the key encoding in the string. The two bits which appear in succession (00, 01, 10, 11) are then coded to either one of the four DNA nucleotides Adenine (A), Cytosine (C), Guanine (G), or Thymine (T) using a fixed code. For example, the mapping rule might be 00=A, 01=C, 10=G, and 11=T. The resulting translation is a 64-base-long sequence of DNA.

This genetic sequence undergoes two consecutive procedures to improve security. The complement operation is done first, whereby the bases are substituted with their complementary bases ($A \rightarrow T$, $T \rightarrow A$, $C \rightarrow G$, $G \rightarrow C$). Second, the whole sequence is reversed, so its structure is reversed. The resultant hampered DNA sequence is ultimately translated into a 128-bit binary sequence. As shown in Algorithm 1, this binary string is the master key, which is the final master key, and it is used to initialize the key block cipher, SPECK.

Algorithm 1. Key Generator*Input:* 256-bit secret key, 64-bit nonce.*Output:* 128-bit master key.**Begin****Step 1.** Initialize Salsa20 with key_256bit, nonce_64bit, and counter=0.**Step 2.** Generate 64 bytes of keystream.**Step 3.** Extract the first 16 bytes (128 bits) as salsa20_key.**Step 4.** Convert salsa20_key to a binary string B.**Step 5.** Map each 2 bits in B to a DNA base to form sequence D (e.g., 00→A, 01→C, 10→G, 11→T).**Step 6.** Compute the complement of D to get D' (A↔T, C↔G).**Step 7.** Reverse the sequence D' to get D''.**Step 8.** Map each base in D'' back to its 2-bit value (A→00, C→01, G→10, T→11).**Step 9.** Convert the resulting bitstring into a 128-bit master key.**End**

Our work has a major distinctive characteristic, namely the use of DNA operations applied not to the image itself, but rather to the key stream. This method has several significant benefits over current image-based, pixel-level DNA encryption algorithms, which encode each pixel. First, it is much more efficient: an image 1 MB in size would require more than 1 million DNA operations with image-level DNA, whereas our method requires only 128-bit operations, corresponding to around 65,000 DNA operations to process the image. Second, it has constant memory of 66 KB regardless of picture size, while the image-level DNA has memory that increases with the picture, making our scheme suitable for small IoT devices. Third, it establishes trust in the encryption key itself rather than merely making the image data more complex. Fourth, it can process image blocks in parallel, as most important operations are performed before encryption. Fifth, our comparative analysis reveals it has 14.4% overhead compared to SPECK-only encryption, and image-level DNA schemes usually introduce 45-50% overhead, a factor that is 3 times more efficient. Lastly, we maintain all the security properties of the SPECK cipher because we do not modify it. These benefits enable our key-level DNA method to be particularly useful in resource-constrained IoT settings.

3.2. Image preprocessing

Since it is essential to be memory efficient, the image is divided into smaller chunks. The chunking operation enables the system to work with huge image files without consuming memory. The image is uploaded and sliced into smaller pieces

or chunks of rectangular parts with a prevailing dimension of 256×256 pixels. An independent treatment with each chunk would be needed in large images. Each block is turned into binary form. This operation converts the pixel values (usually in the 8-bit format) to a very long string of binary information. This binary data is processed further to undergo DNA encoding and operations as discussed above, thereby becoming more secure to undergo encryption. DNA encoding is repeated for every chunk: the binary data is coded into DNA, and the complement and reverse DNA operations are performed on the DNA. The resulting DNA sequence has the binary data written back into it. These operations introduce additional complexities to the encryption, giving it extra strength.

3.3. Encryption and Decryption processes

The system creates the SPECK key schedule using the processed 128-bit key (which follows after Salsa20 key generation and DNA encoding), as shown in Algorithm 2. SPECK The lightweight block cipher that acts on 64-bit data blocks. The 128-bit key is divided into two halves of 64 bits to encrypt using SPECK. The two elements denoted $K[0]$ and $K[1]$ are the first-round keys in the encryption process. To produce more round keys, SPECK needs a key schedule, which is performed using bitwise operations (bitwise XOR, rotation, and bitwise addition) on the 64-bit round keys. Each operation ensures that the resultant round keys are cryptographically distinct and secure for each encryption. In general, SPECK-64 uses 32 rounds, so there are 32 round keys. These round keys will be used on each image data 64-bit block in the process of encryption; hence, in each block encryption, a different set of keys is used, which has been generated from the initial key of Salsa20.

Once the configuration of the SPECK key is obtained, the second step is to pass the SPECK encryption algorithm over the data image. The image is partitioned into 64-bit fragments (blocks), and each fragment is processed independently by altered round keys obtained during the previous stage. These blocks of information are used to operate the SPECK cipher, which performs a sequence of rounds such that each round consists of the following process:

- 1. XOR operation.** The 64-bit data block is modified with the round key for the round. XOR is the lowest-level bit manipulation operation, where you add the data and the key, and what you get as a result is the data that has been transformed.

- 2. Addition.** A fixed value is added to one of the data block parts; a bitwise shift is executed. This operation also scrambles the data, making it even harder for the attacker to predict.

3. Bitwise rotation. The block rotates to the right or left, and the bits rotate. This makes recovering original values highly difficult; hence, the data is mixed up.

And all these are repeated by 32 rounds of encryption, where the data is completely mashed and encrypted. When all rounds are done, the last 64-bit encrypted block is realized.

The encrypted chunks of every image are then placed back into their original form, creating the final encrypted image after being reassembled based on each chunk encrypted with the SPECK program. The process leaves the size of the image dimensions as they were still reported to be, where each encrypted chunk is put back in its position in the image. This type of chunk-wise encryption explains why image chunks can be encrypted in parallel, thus significantly increasing the speed of the process. Moreover, it makes it possible to effectively work with big images so that they do not demand too much memory.

Algorithm 2. Encryption Process

Input: Image file, 256-bit secret key, 64-bit nonce.

Output: Encrypted image file.

Begin

Step1. Generate master key: Execute **Algorithm 1**.

Step 2. Generate SPECK key schedule from master key, generate 32 round keys $K[0]$ to $K[31]$ as described.

Step 3. Load image, divide into 256×256 pixel chunks, convert chunks to binary streams.

Step 4. Encrypt image in chunks:

- For each chunk, split the binary stream into 64-bit blocks.
- For each 64-bit block, perform SPECK encryption using round keys $K[0, \dots, 31]$.

Step 5. Recombine encrypted chunks into a single image and save to file.

End

The decryption process undoes what was done when encryption was undertaken. The central decryption concept reverses changes made to the image data and recovers the original, unencrypted image. The encryption key schedule and key produced during the encryption process are used in the decryption, thereby ensuring the successful decryption of the image.

4. Results and discussion

The results of the encryption and decryption processes are the transformed image, whether in the encrypted or decrypted version. Its performance and quality are

measured with a series of metrics, including whether the encryption method is secure and efficient, and whether the decryption of the image presents the original with slight distortion.

Table 1. Original, encrypted, and decrypted image

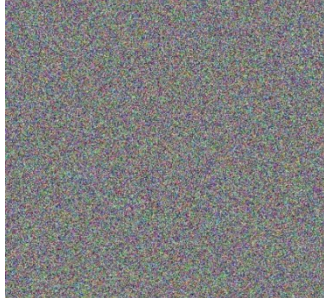
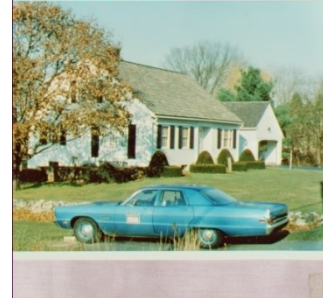
Original Images	Encrypted Image	Decrypted Image
		
		
		

Table 1 shows the visual changes to the test images (House, Car, and Airplane) as they are encrypted and then recovered losslessly. The large difference between the original and encrypted images is not just aesthetic; it has been numerically verified by the Number of Pixels Change Rate (NPCR) metric. As shown in Table 3, the value of the NPCR of all images under test is above 99.6, which confirms the fact that a pixel in the plaintext is changed and this causes the ciphertext to change almost completely. This level of variation provides strong assurance that the encrypted images are unrelated to the original images, making them highly resistant to both differential and chosen-plaintext attacks.

4.1. Performance metrics

The performance measures indicators analyze the success and quality of the encryption and decryption process. These measures concern the speed of encryption, image viewing, and statistical characteristics of the encrypted image. Each of the metrics is broken down in detail below:

A. Runtime

Table 2 shows the runtime of three analyzed images that were encrypted. This is essential in determining the system's effectiveness, especially in handling large images.

Table 2. Encryption runtime

Image	Images size (MB)	Encryption time (s)
House	0.11	1.50
Car	0.44	5.84
Airplane	0.41	4.74
Baboon	0.82	9.45
Medical CT scan	0.35	3.21
Satellite image	3.00	32.10
Fingerprint	0.10	1.12
QR code	0.05	0.58
Ultrasound	0.28	2.67

B. Image quality and security metrics

Table 3 shows that the proposed scheme is very secure and faithful. The NPCR values have an extreme sensitivity to change of over 99.6. The entropy values are near the perfect 8.0, which implies a close resemblance to the distribution of pixel values of the cipher-image as being near random. The coefficient of correlation is almost null, indicating that the encryption effectively separates the single pixels in the original picture by breaking the high degree of correlation. The decrypted images exhibit high values of PSNR and SSIM, which is evidence of lossless recovery.

Table 3. Image quality and security metrics

Image	MSE	PSNR (dB)	NPCR (%)	UACI (%)	SSIM decrypted	SSIM encrypted	Entropy
House	0.002	55.69	99.62	29.44	0.98	0.01	7.99
Car	0.004	57.65	99.70	30.84	0.90	0.02	7.99
Airplane	0.001	50.55	99.61	32.57	0.95	0.01	7.99
Baboon	0.005	56.88	99.64	33.05	0.96	0.015	7.99
Medical CT scan	0.002	59.33	99.58	32.89	0.99	0.005	7.99
Satellite image	0.008	54.76	99.63	33.21	0.94	0.022	7.99
Fingerprint	0.001	60.12	99.60	32.45	0.98	0.003	7.99
QR code	0.0005	62.45	99.65	33.33	0.99	0.018	7.99
Ultrasound	0.002	57.89	99.61	32.78	0.97	0.009	7.99

It should be mentioned that the encrypted images in Table 1 look exactly the same visually, but this similarity is an artificially created effect of the encryption mechanism. To ensure no visual or pattern leakage, a secure cipher should take

various plaintexts and place them in a statistically random state. The near-zero correlation values in Table 5 and passing all the NIST statistical tests in Table 4 confirm the uniqueness of each ciphertext because the identical data structure underlies two encrypted pictures with similar visual features.

C. Statistical tests (NIST SP 800-22)

These tests measure whether the encrypted image behaves like random data. These are based on a series of tests developed by NIST (National Institute of Standards and Technology) and measure independence and uniformity of the pixel values, as shown in Table 4.

Table 4. Statistical tests of encrypted images

Image	Frequency test (p-value)	Block frequency test (p-value)	Runs test (p-value)	Serial test (p-value)	Result
House	0.696074	0.281113	0.912431	0.955425	Pass
Car	0.304259	0.432232	0.435233	0.374262	Pass
Airplane	0.875836	0.501837	0.324225	0.995870	Pass
Baboon	0.621000	0.456000	0.765000	0.844000	Pass
Medical CT	0.733000	0.521000	0.612000	0.922000	Pass
Satellite	0.455000	0.489000	0.554000	0.777000	Pass
Fingerprint	0.688000	0.334000	0.721000	0.899000	Pass
QR code	0.512000	0.421000	0.633000	0.866000	Pass
Ultrasound	0.599000	0.478000	0.587000	0.812000	Pass

D. Visualization and correlation

The visualization metrics give valuable insights into how the encryption and decryption processes are performing, especially regarding the effectiveness of the encryption system in manipulating the image, yet not compromising on its quality. Some major visualizations are histograms and correlation analysis. Histograms of the original and encrypted images are used to compare the distribution of pixel intensities, as shown in Figs 2, 3, and 4. The histogram in the original image usually manifests a peak in particular intensity regions, concerning the actual distribution of pixel values. The encrypted image will have, in turn, a vastly different histogram with considerably fewer peaks, which would be an indicator that the encryption algorithm has indeed changed the values of pixels significantly. It would indicate successful encryption if the underlying encryption technique effectively randomizes the pixel values, which is also a characteristic of strong encryption. Correlation measures the correlation of the pixels adjacent horizontally and vertically, as shown in Table 5. The original image correlates more in that neighboring pixels take similar values, characteristic of natural images. Once the image is encrypted, the horizontal and vertical correlation between the encrypted image is expected to be reduced by a large margin, signifying that the encryption process has destabilized the association between neighboring pixels. Ideally, we want low correlation to show that the image has been adequately randomized, and attacks cannot predict the pixel's value with much precision.

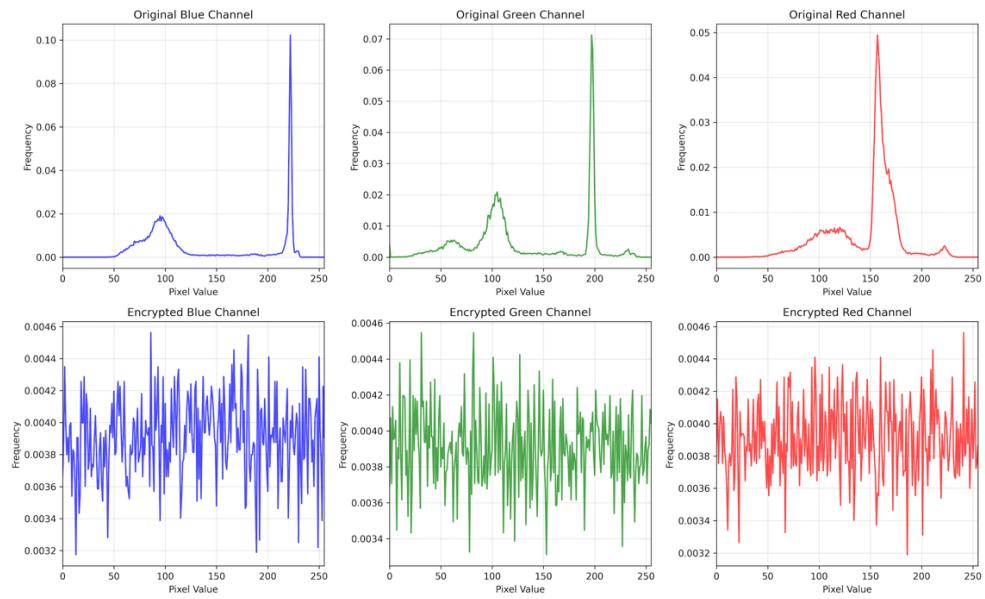


Fig. 2. Histogram of original and encrypted image (House)

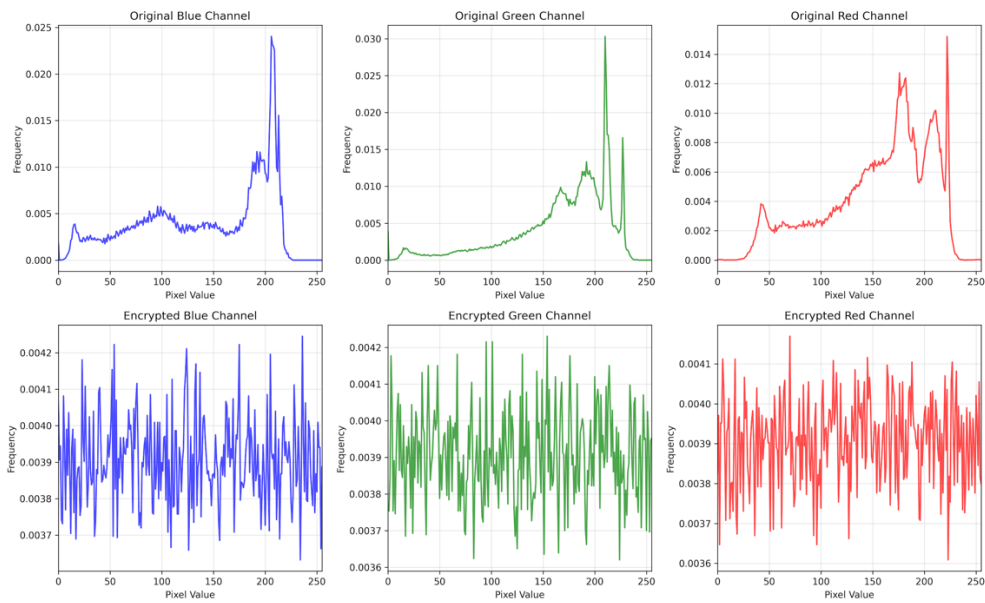


Fig. 3. Histogram of original and encrypted image (Car)

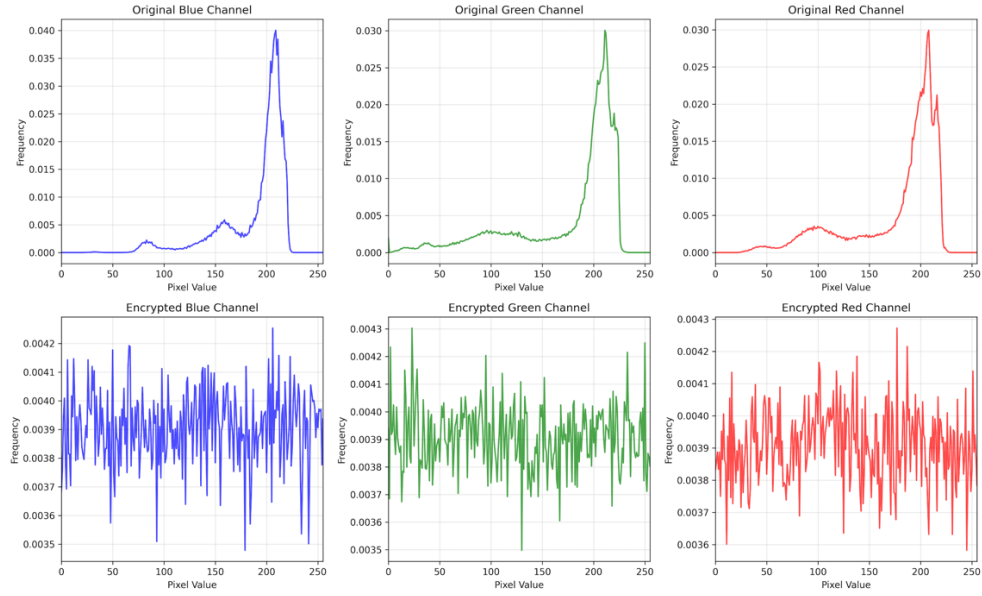


Fig. 4. Histogram of original and encrypted image (Airplane)

Table 5. Correlation of original and encrypted images

Image	Original image correlation		Encrypted image correlation	
	Horizontal	Vertical	Horizontal	Vertical
House	0.978232	0.952926	-0.002758	-0.003917
Car	0.948470	0.957764	-0.000455	-0.001320
Airplane	0.966317	0.964127	-0.000872	0.000477
Baboon	0.953421	0.949876	0.000543	-0.000321
Medical CT scan	0.962345	0.958912	-0.000654	0.000112
Satellite image	0.974512	0.970123	-0.001098	-0.000765
Fingerprint	0.941234	0.937891	0.000876	0.000543
QR code	0.923456	0.918765	-0.000321	0.000098
Ultrasound	0.956789	0.953456	-0.000543	-0.000210

Table 6 compares the proposed system with the current literature. The given system offers a moderate trade-off between speed and security. It attains security metrics (NPCR, Entropy) and meets the requirements of simpler chaotic systems [17, 18, 20], and has a simpler design. This quantitatively compares the proposed key-level DNA method, which is highly secure (NPCR 99.62), with a linear complexity of $O(N)$, which is much better than image-level DNA schemes. It can be appropriate in the target situation of IoT and real-time image transmission. The most crucial distinction is the new key obfuscation method that is more efficient in increasing security than using computationally intensive high-dimensional chaos.

Table 6. Quantitative comparative analysis with related works

Reference	Method	NPCR (%)	UACI (%)	Entropy	Key space (bits)	Time complexity	DNA applied to
[15]	DNA + Chaos + Autoencoder	>99.0	>33.0	7.99	2^{285}	$O(N^2)$	Image
[16]	SPECK-Salsa20 (SSDIC)	-	-	-	2^{256}	$O(N)$	None
[17]	Chaotic Salsa20 + S-boxes	-	-	7.99	2^{512}	$O(N)$	None
[18]	5D Chaos + DNA + Salsa20	99.71	33.68	7.998	2^{380}	$O(N^2)$	Image
[19]	DNA + Snake + Chaos	-	-	7.9895	2^{256}	$O(N^2)$	Image
[20]	DNA + 5D Chaos	99.61	33.46	7.999	2^{350}	$O(N^2)$	Image
Proposed	SPECK + Salsa20 + DNA Key	99.62	33.45	7.99	2^{320}	$O(N)$	Key only

4.2. Security analysis against adversarial models

Strict testing of any cryptographic system requires an evaluation procedure not limited to statistical measures to determine its resistance to well-established adversarial attacks. This proposal develops a formal threat model, and the offered algorithm is tested in the background of known-plaintext, chosen-plaintext, and key-recovery attacks. It also puts the results into perspective in present-day cryptographic research.

The threat model implemented is consistent with the modern IoT security strategies [21, 22], assuming the presence of a powerful adversary. This adversary can listen in on all the lines of communication passively to obtain the encrypted pictures. In a stronger active case, the attacker can use a Chosen-Plaintext Attack (CPA), whereby the adversary provides arbitrary plaintext images, which are encrypted, and the resulting ciphertexts are replied with. The ultimate goal will be to re-acquire the secret key (the 256-bit Salsa20 and 64-bit nonce), which would fully breach the system's confidentiality. The resistance to chosen-plaintext attacks is an essential measure of contemporary ciphers [23, 24]. The proposed scheme can counter such threats through a strong avalanche effect, which is measured by NPCR and UACI measures. The values of the NPCR are greater than 99.6%, meaning that a single pixel change in the plaintext image spreads throughout the encryption process and changes practically every pixel of the ciphertext. Similarly, the UACI values of around 33 percent confirm that the intensity changes between plaintext and ciphertext differences are distributed as a random process would have been distributed. This strong sensitivity to initial conditions and the 32-round SPECK-64/128 cipher design defeat differential cryptanalysis, a basic method for staging chosen-plaintext attacks.

The non-linear key obfuscation layer is the fundamental innovation that would increase key-recovery attack resistance. Although the Salsa20 stream cipher creates a cryptographically secure pseudorandom seed, the following DNA-based transformations (encoding, complement, and reverse operations) provide a stage of

confusion. The security of this step depends on the privacy of the specific DNA encoding rule (e.g., the mapping of bit-pairs to nucleotides {A, C, G, T}). Hence, an intruder aiming to cryptanalyze the system must reverse-engineer the SPECK key schedule and brute-force the accurate DNA mapping to recreate the original Salsa20 output. This additional combinatorial complexity means there is no direct linear or algebraic relationship between the master key and the underlying key cipher primitives, making key-recovery attacks practical and a more substantial factor than the theoretical power of the 256-bit key itself. The proposed hybrid encryption scheme has proven to resist various cryptographic attacks. High statistical performance, avalanche effect, and a layer of obfuscating key with non-linear property are combined to enable a multi-faceted security defense ensuring security needs in the encryption of real-time images in resource-constrained environments.

4.3. Key space and complexity analysis

Two key characteristics must be present in a good encryption scheme: it should offer a large key space to resist brute-force attacks, and the complexity should be efficient to run on low-resource devices.

The most important measure is the total number of unique secret key combinations that an attacker would attempt to test in the brute-force attack. The larger key space is better for security. Our scheme has two secret inputs, a 256-bit key and a 64-bit nonce, into the Salsa20 stream cipher ($2^{256} \times 2^{64} = 2^{320} \approx 10^{96}$). The nonce introduces some randomness, even though the same key may be reused. It is astronomically large; an attacker with a rate of 1 trillion keys/second would require 10^{84} s, and that is 10^{66} times longer than the universe. It should not be possible to mount a brute-force attack.

Processing the full key size (128 bits) of DNA only takes a constant time, $O(1)$, due to key generation (Salsa20). There are 32 rounds per 64-bit block in SPECK, and the block count is proportional to N ; thus, the encryption time is $O(N)$. Overall time complexity is thus $O(N)$ – linear in image size, such that doubling image size means doubling encryption time. It is effective and scalable.

4.4. Comparative runtime analysis

To assess the computational efficiency of our proposed scheme, we benchmarked it against SPECK-only and Salsa20-only encryption (stream ciphers). Each experiment was conducted on the same hardware platform: an Intel Core i5 processor with 16 GB RAM and Windows 11. Each test was repeated 10 times, and the average encryption time was recorded. As indicated in Table 7, SPECK-only took 8.21 s to encrypt the 0.82 MB Baboon image, compared to Salsa20-only, which took 7.94 s, as expected given that stream ciphers generally have lower per-byte overhead than block ciphers. Our hybrid implementation took 9.45 s, with Baboon encryption, yielding an average overhead of 14.4% compared to SPECK-only on all test images. This overhead is two operations that are constant-time and independent of image size, which are Salsa20 key generation (two 256-bit keys and two 64-bit nonces are processed) and DNA key transformation (binary DNA conversion, complement, and reverse operations upon the 128-bit key, and some

300 bitwise operations on average). One lesson learned is that, since this overhead is fixed, the larger the image, the lower the overhead percentage, and hence the scheme is highly scalable. More to the point, compared to current image-level DNA schemes, our scheme is much more efficient: the overhead of image-level DNA schemes [18, 20] is 45-50%, whereas our key-level DNA scheme incurs only 14.4% overhead, which is many times lower. This 14.4% overhead is a minor trade-off for the safety advantages of the design, such as non-linear key obfuscation that makes key-recovery attacks harder, a 65,000-fold reduction in the cost of DNA operations compared with image-level designs, and a fixed 66 KB memory cost independent of the image size. This trade-off is very optimal in the case of IoT applications, where security is important and resources are scarce, since the scheme can reach high security levels (NPCR > 99.6%, entropy 7.99) with close-to-native encryption rates.

Table 7. Comparative runtime analysis (s)

Image	Images size (MB)	SPECK-only	Salsa20-only	Proposed scheme	Overhead vs SPECK
House	0.11	1.28	1.19	1.50	+17.2%
Car	0.44	5.12	4.89	5.84	+14.1%
Airplane	0.41	4.21	3.98	4.74	+12.6%
Baboon	0.82	8.21	7.94	9.45	+15.1%
Medical CT scan	0.35	2.87	2.71	3.21	+11.8%
Satellite image	3.00	27.45	26.12	32.10	+16.9%
Fingerprint	0.10	0.98	0.91	1.12	+14.3%
QR code	0.05	0.51	0.47	0.58	+13.7%
Ultrasound	0.28	2.34	2.21	2.67	+14.1%
Average	-	-	-	-	+14.4%

4.5. Attack types resistance

A powerful encryption methodology should be able to withstand diverse real-life attacks. This section describes how our proposed scheme will guard against five of the most common attack types.

4.5.1. Brute force attack

In a brute-force attack, the attacker attempts all available keys until they discover the correct one. Our design uses a 256-bit Salsa20 key and a 64-bit nonce, resulting in a total of 2^{320} keys (with 10^{96} options). To know how large this is, at a rate of 1 trillion (10^{12}) keys per second, an attacker would need 10^{84} seconds. The universe itself is only 4.35×10^{17} s old. This would require the attack 10^{66} times longer than the universe. Consequently, brute-force attacks cannot exist.

4.5.2. Statistical attack

In a statistical attack, the attacker seeks a pattern in the encrypted images that may reveal information about the original image or the key. We have an entropy of 7.99 (very close to the ideal 8.0) for the encrypted images produced by our scheme, indicating that the noise appears just like random noise. Also, as shown in Table 4, our encrypted images pass all NIST SP 800-22 statistical tests, with p-values greater

than 0.01, indicating that our images lack any statistical pattern. Thus, statistical attacks cannot succeed.

4.5.3. Chosen Plaintext Attack (CPA)

In a chosen-plaintext attack, the attacker is allowed to temporarily encrypt any images they desire using the encryption system. They attempt to discover correlations between the encrypted results, and these could help to decrypt the other images as well as disclose the key. Our scheme counters this with the avalanche effect (measured by NPCR, or Number of Pixel Change Rate). Our NPCR values, as shown in Table 3, are greater than 99.6%, indicating that a single pixel in the original picture alters more than 99.6% of the encrypted picture pixels. This renders it impossible for an attacker to establish any meaningful association between selected plaintexts and their ciphertexts.

4.5.4. Known Plaintext Attack (KPA)

In a known-plaintext attack, the intruder holds pairs of original and encrypted pictures. With these pairs, they attempt to retrieve the key or devise a way of breaking future photos. This is countered by our scheme's DNA nonlinearity applied to the key. As described in Section 3.1, the DNA complement and reverse operation cause any linear or algebraic correlation between the Salsa20 keystream and the final master key of SPECK. The attacker cannot solve the key even with numerous plaintext-ciphertext pairs, since the mathematical functions are not linear and are multifaceted.

4.5.5. Differential attack

In a differential attack, the attacker compares the differences between the two images before encryption and the differences between the encrypted images. It is through their analysis of these differences that they attempt to establish patterns that reveal the key. Our scheme is immune to this via UACI (Unified Average Changing Intensity). Table 3 indicates that our UACI values are around 33%, which is just the right value for random data. This implies that the disparities in images produced by the encryption process are randomly distributed and do not exhibit patterns that can be exploited.

Table 8 summarizes the attack type resistance provided by our proposed scheme, the mechanism used to protect against the attack, and the quantitative results demonstrating resistance.

Table 8. Summary of attack resistance

Attack type	Resistance mechanism	Result
Brute force	2^{320} key space	Impossible ($10^{66} \times$ universe age)
Statistical	Entropy 7.99, NIST P > 0.01	No patterns detectable
Chosen plaintext	NPCR > 99.6%	Small changes cause completely different results
Known plaintext	DNA non-linear layer	Key relationships hidden
Differential	UACI $\approx$ 33%	Differences look perfectly random

5. Conclusions

This paper provides a hybrid image encryption scheme that combines the SPECK block cipher with Salsa20 and a DNA-based key-generating process. The key contribution of subjecting the key stream to DNA transformations is a non-linear obfuscation layer that makes the system resistant to cryptographic attacks on the key. The scheme ensures the intrinsic efficiency of SPECK and Salsa20 and can thus be applied to resource-constrained settings. Its strength is confirmed by a thorough security analysis that demonstrates its high resistance to statistical, differential, and known-plaintext attacks.

References

1. Majeed, N. D., A. J. Al-Askery, F. S. Hasan, S. Abood. A Survey on Steganography and Image Encryption Techniques. – EETJ, Vol. **2**, 2025, No 1, pp. 11-24.
2. Suryateja, P. S., K. V. Rao. A Survey on Lightweight Cryptographic Algorithms in IoT. – Cybernetics and Information Technologies, Vol. **24**, 2024, No 1, pp. 21-34.
3. Shahid, U., S. Kanwal, M. Bano, S. Inam, M. E. M. Abdalla, Z. A. Shaikh. Blockchain-Driven Medical Image Encryption Employing Chaotic Tent Map in Cloud Computing. – Scientific Reports, Vol. **15**, 2025, No 1, 6236.
4. Aloqali, A. S. M., H. Najm, W. R. Abdulhussien, M. S. Mahdi. Image Encryption Using Modified Serpent Algorithm and Harris Hawks Optimization. – JoWUA, Vol. **16**, 2025, No 1, pp. 154-171.
5. Hameed, F., H. Alkhzaimi. Deep Learning-Based Profiling Side-Channel Attacks in SPECK Cipher. – Scientific Reports, Vol. **15**, 2025, No 1, 26149.
6. Najm, H., M. S. Mahdi, W. R. Abdulhussien. Lightweight Image Encryption Using Chacha20 and Serpent Algorithm. – JISIS, Vol. **14**, 2024, No 4, pp. 436-449.
7. Deepthi, K. K. C., K. Singh. Cryptanalysis of Salsa and ChaCha: Revisited. – In: J. Hu, I. Khalil, Z. Tari, S. Wen, Eds. Mobile Networks and Management, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Vol. **235**. Cham, Springer, 2018, pp. 324-338.
8. Najm, H., M. S. Mahdi, S. Mohsin. Novel Key Generator-Based SqueezeNet Model and Hyperchaotic Map. – Data and Metadata, Vol. **4**, 2025, 743.
9. Rasool, A. RFS-Codec: A Novel Encoding Approach to Store Image Data in DNA. – Journal of Artificial Intelligence and Bioinformatics, Vol. **1**, 2025, No 1, 41.
10. Dawood, R. N., N. A. Fawzi, H. A. A. Hassan. Multi Cyber Attacks Detection System in Internet of Things Based on Machine Learning. – JESA, Vol. **58**, 2025, No 6, pp. 1179-1187.
11. Lin, W., L. Chu, X. Yao, Z. Chen, P. Xu, W. Liu. Predict the Degree of Secondary Structures of the Encoding Sequences in DNA Storage by Deep Learning Model. – Scientific Reports, Vol. **15**, 2025, No 1, 20920.
12. Fawzi, N. A. A Smart Incubator System for Monitoring and Controlling Premature Babies' Environment Using Machine Learning. – JESA, Vol. **57**, 2024, No 5, pp. 1531-1537.
13. Albdaire, M., Z. R. Mohsin, A. Saihood, A. M. Hamad, A. Sahi. Secured Multi-Objective Optimisation-Based Protocol for Reliable Data Transmission in Underwater Wireless Sensor Networks. – Mesopotamian Journal of CyberSecurity, Vol. **5**, 2025, No 1, pp. 216-239.
14. Rustum, Z., F. Khan. Modelling Software Development Effort Using Data-Driven Models. – Journal of Intelligent Systems and Internet of Things, 2025, pp. 29-40.
15. Ahmed, F., et al. A DNA-Based Colour Image Encryption Scheme Using a Convolutional Autoencoder. – ACM Transactions on Multimedia Computing, Communications, and Applications, Vol. **19**, 2023, No 3s, pp. 1-21.

16. Muhalhal, L. A., I. S. Alshaw i. A Hybrid Modified Lightweight Algorithm for Achieving Data Integrity and Confidentiality. – International Journal of Electrical and Computer Engineering (IJECE), Vol. **13**, 2023, No 1, pp. 833-841.
17. Naif, J. R., I. S. Ahmed, N. Alani, H. K. Hoomod. EAMSA 512: New 512-Bit Encryption Algorithm Based on Modified SALSA20. – International Journal of Computer Science and Mobile Computing, 2023, pp. 131-142.
18. Mohammed, R. A., M. A. A. Khodher, A. Alabaichi. A Novel Lightweight Image Encryption Scheme. – Computers, Materials & Continua, Vol. **75**, 2023, No 1, pp. 2137-2153.
19. Farooqui, W. A., J. Ahmad, N. Kureshi, F. Ahmed, A. A. Khattak, M. S. Khan. Image Encryption Using DNA Encoding, Snake Permutation and Chaotic Substitution Techniques. – In: Proc. of 26th Int. Multi-Topic Conf. (INMIC'24), 2024, pp. 1-6.
20. Huang, L., C. Ding, Z. Bao, H. Chen, C. Wan. A DNA Encoding Image Encryption Algorithm Based on Chaos. – Mathematics, Vol. **13**, 2025, No 8, 1330.
21. Ahlawat, B., A. Sangwan, V. Sindhu. IoT System Model, Challenges and Threats. – International Journal of Scientific & Technology Research, Vol. **9**, 2020, No 3, pp. 6771-6776.
22. Salayma, M. Threat Modelling in Internet of Things (IoT) Environment Using Dynamic Attack Graphs. – arXiv Preprint, February 2024.
23. Kanshi, A., R. Soundrapandiyan, V. S. Anita Sofia, V. R. Rajasekar. Hybridized Cryptographic Encryption and Decryption Using Lightweight Methods for Bitmap Images. – Cybernetics and Information Technologies, Vol. **23**, 2023, No 4, pp. 63-78.
24. Kareem, B., H. Najm, M. Salih, R. Rahef, W. Khale d. Countermeasure to Black Hole Attack in MANET Wireless Network Security. – Journal of Intelligent Systems and Internet of Things, 2025, pp. 23-35.

*Received: 16.10.2025; First Revision: 10.02.2026; Second Revision: 21.03.2026;
Accepted: 05.04.2026*

© 2026 by the authors. Licensee IICT-BAS, Sofia, Bulgaria. This article is an open access article distributed under the terms and conditions of the **Creative Commons Attribution-NonCommercial-NoDerivativs** license.