The object of the study is the AES-128 (Advanced Encryption Standard) – based image-encryption scheme. The problem solved is the persistence of residual image structure and sub-ideal statistical security when classical AES is naively applied to visual data. The generated cipher images yield near-maximal entropy with low pixel correlations and uniform histograms. Encrypted-image Chi-square values concentrate around 200–310 (close to a uniform distribution), the NPCR (Number Of Changing Pixel Rate) consistently 99.623–99.657% with a best case 99.6547%, and the UACI (Unified Averaged Changed Intensity) ≈ 33.64% per channel (RGB combined ≈ 22%). Robustness tests show ≈ 30.7 dB at 50% cropping and ≈ 39.5–39.7 dB at 0.01 salt-and-pepper noise and 6.25% cropping. These outcomes are explained by the bit-level, state-dependent permutations introduced by the surjective automaton (boosting diffusion) and by the nonlinear S-box synthesized under strict criteria (e. g., bounded differential uniformity, high nonlinearity) that heighten confusion, and operation in CBC (Cipher Block Chaining) mode supplies semantic security. Other unique features that facilitate the solution are the substituting of ShiftRows/MixColumns with surjective finite automata; a custom, criteria-optimized S-box; and a 10-round AES-128 CBC pipeline with a random. Taking all of this together yielding observed statistical uniformity, a high NPCR/UACI, and stable robustness under degradation. Lastly, the findings demonstrate the applicability to secure multimedia transmission and storage in channels prone to noise or partial data loss, and being data-agnostic, that the transformations can generalize to text and generic binary data when carefully managed

Keywords: image encryption, surjective finite automaton, custom S-box, AES

UDC 004.056.55
DOI: 10.15587/1729-4061.2025.348368

DEVELOPMENT OF IMAGE ENCRYPTION METHOD USING SURJECTIVE FINITE AUTOMATA AND CUSTOM S-BOX WITHIN THE ADVANCED ENCRYPTION STANDARD FRAMEWORK

Alibek Barlybayev
Doctor PhD, Professor*
Zhanat Saukhanova
Candidate of Physical and Mathematical Sciences, Associate Professor**
Gulmira Shakhmetova
Corresponding author
Senior Lecturer**
E-mail: sh_mira2004@mail.ru
Altynbek Sharipbay
Doctor of Technical Sciences, Professor*
Sayat Raykul
Student**
Altay Khassenov
Student**
*Department of Artificial Intelligence Technologies***
Department of Information Security*
***L.N. Gumilyov Eurasian National University
Satbaev str., 2, Astana, Republic of Kazakhstan, 010000

Received 04.10.2025
Received in revised form 27.11.2025
Accepted date 17.12.2025
Published date 30.12.2025

How to Cite: *Barlybayev, A., Saukhanova, Z., Shakhmetova, G., Sharipbay, A., Raykul, S., Khassenov, A. (2025). Development of image encryption method using surjective finite automata and custom S-box within the advanced encryption standard framework. Eastern-European Journal of Enterprise Technologies, 6 (9 (138)), 77–99. https://doi.org/10.15587/1729-4061.2025.348368*

1. Introduction

The need for methods is still strong because digital technology changes fast and new attackers get smarter. The modern cryptographic systems face threat models that old cryptographic schemes did not plan for old cryptographic schemes have holes when users deploy old cryptographic schemes today [1]. The new attacks keep coming. The attacks include channel tricks, adaptive attackers and large automated cryptanalysis. The research into methods is needed to make digital security stronger [2]. The value of this topic grows because the world depends on communication. Today online banking, e-commerce, government services, industrial control systems and everyday messaging all need protection. If confidentiality fails cryptographic protection fails. If integrity fails cryptographic protection fails. If non-repudiation fails cryptographic protection fails. Any of those failures can cause loss, loss of privacy or a break, in critical infrastructure. In this study put the state machines inside the cipher pipeline and rethink the encryption and the decryption.

Finite-state machines (FSMs) give a flexible base, for the cipher designs [3]. Finite-state machines (FSMs) handle the encryption and the decryption in a way that's not like the block and stream ciphers. The block and stream ciphers are under pressure, from the analysis methods. In this work use computation and controlled reversibility [4]. Using computation and controlled reversibility build a communication scheme. The confidential communication scheme also adds the integrity and the non-repudiation. This research relies on automata theory. This study has focus on Mealy automata and surjective automata. Also examine Mealy automata and surjective automata to improve the resistance to code breaking and to ensure the reversal.

Secure channels matter. Secure channels are becoming more important, in shopping, banking and everyday digital

messaging. A leak can cause damage anywhere. Study finite automata in this context to open directions for theory and practice in computer science and information security. Aim to build and test an automata-based cryptosystem and to show that the automata-based cryptosystem works in the conditions not just on paper. Have a longer-term goal to help set a standard for cryptography, as digital technologies and attack methods continue to develop.

Traditional algorithms have limits when they process the content. In this paper present a state machine-based cipher. Test the cipher with experiments. After review the literature and the theory implement the proposed method. Then run experiments on the test images. Measure quality and security, with the pixel-correlation reduction the entropy, the NPCR/UACI, the square goodness-of-fit and the resistance to occlusion and injected noise. Compare the cipher with the AES and the DES to get a reference point, for robustness. Finish with observations, on the bottlenecks. Observations on the bottlenecks lead to outline the steps, for the optimization and the hardware mapping.

The rise of cyber threats limits the ciphers for image data and the clear benefits of state machines with state dependent and reversible transformations all appear now. This situation creates a need and a scientific need. The study tests the automaton-based encryption design against metrics such as entropy, correlation, NPCR/UACI, chi square and occlusion noise resistance. The study compares the automaton-based encryption design, to AES and DES. Then it is possible to see that the confidentiality, integrity and non-repudiation needs solutions in risk areas. High risk areas include e-commerce, online banking and digital communications. It is possible to notice that the development of systems based on finite-state machines addresses the confidentiality, integrity and non-repudiation needs. It is possible to notice that the evaluation of systems based on finite-state machines shows why the development matters for the confidentiality, integrity and non-repudiation needs, in risk areas. Therefore, research in the field of developing encryption methods using finite-state machines is relevant.

2. Literature review and problem statement

The paper [3] presents the results of research that treats encryption and decryption as finite-automata (FA) transformations. It is shown that the keys set the states, the transitions and sometimes the shape of the automaton. The keys create a state space. The encryption and decryption then work like substitution-permutation in both streaming and block modes. Researchers assess the security of the encryption and decryption, with period and linear-complexity metrics with indistinguishability tests and with checks for resistance, to attacks and brute-force attacks. This approach gives a view of how encryption and decryption can be built from finite-automata. This approach focuses on flexibility, in ideas while following design rules such as state size, nonlinear transitions and output mixing. FA-centric constructions can be used to shape security profiles for applications. The approach does not yet give guarantees of strong randomness, at practical scales. The approach also does not yet give guarantees of machine-level reversibility at scales. The approach also does not yet give real-time throughput on platforms. The approach also lacks proofs that can resist cryptanalysis. The approach also lacks proofs that can resist quantum cryptanalysis. Let's notice these gaps in the approach.

The paper [5] presents a study that organizes algorithms into symmetric families and asymmetric families. It is shown that each algorithm evaluates by architecture (Feistel, vs. SPN) by size by block size by number of rounds by origin by time period by flexibility and by known vulnerabilities. The paper maps the algorithms to attack classes such as force, slide or related-key attacks, differential variants, interpolation attacks, MITM attacks and quantum threats. The taxonomy, in the paper shows which structural decisions lead to weaknesses. It is possible to see that the universal susceptibility to brute-force attacks shows, up everywhere. Lightweight IoT designs must integrate encryption and key exchange tightly. However, unresolved issues remained. First, unification of FA-based design principles with a standardized, attack-oriented evaluation beyond aggregate randomness tests was not achieved. Second, performance parity with optimized baseline AES implementations under identical hardware and I/O constraints was not achieved. Furthermore, the challenge of combining taxonomy-level ideas with constructive, provably reversible automata capable of scaling without prohibitive growth in the number of states remained open. This may be due to objective difficulties. First, difficulties arise during the synthesis or inversion of an automaton due to the combinatorial growth of the state space. Second, a significant barrier is the high cost of reproducible comparative analysis with highly optimized benchmark ciphers. Furthermore, there is a fundamental gap between statistical tests and proofs targeting specific attack models, such as related-key attacks or quantum algorithms. A practical way forward is to ensure surjectivity at the automaton level. Another important step is to embed FA modules into the AES-class permutation-diffusion (SPN) pipeline. Additionally, a reproducible system that produces NIST/ENT estimates along with attack-focused analysis and calibrated throughput and latency on modern hardware is required. Elements of this idea appear in previous works based on FA. However, they are not presented within a unified framework that integrates formal reversibility, standardized metrics, and benchmarks for comparable systems.

The paper [6] presents the results of research in switching and automata theory. The work formalizes deterministic and nondeterministic finite automata. It is shown that how algorithmic state automata (ASM) enable hardware-level specification at the register transfer level. These models are shown to not only capture the behavior of a digital system. They also delineate the boundaries of computability and the complexities associated with security engineering, including decidability issues and the P versus NP problem. They thus provide a rigorous foundation for the design of cryptographic mechanisms. It has been shown that finite-state machine abstractions can be operationalized in cryptography. In particular, keys can be bound to states, transitions, or outputs of the machine. Reversibility can be justified through the machine structure, for example, Mealy or Moore-type. This approach motivates FA-centric cipher designs and authenticated schemes. Several factors contribute to this. First, there are complex engineering obstacles in synthesizing and inverting automata, including state space destruction. Second, the high cost of reproducible benchmarks under identical I/O and hardware conditions is a significant challenge. Furthermore, there is a methodological gap between empirical test suites and proofs tailored to specific adversary models. One way to overcome these difficulties could be to ensure the absence of information loss at the machine level,

i.e., surjectivity. Providing constructive inverses that allow data recovery is also important. Furthermore, FA modules can be embedded in a permutation-diffusion pipeline. This approach should be accompanied by a reproducible test suite (NIST/ENT, NPCR/UACI/χ^2/entropy, PSNR) and attack-oriented analysis.

A growing number of cryptographic schemes are built on finite automata. This paper is being considered of Crypto-Automata on cryptosystems like FAPCK, DAFA, FSAaCIT, FASKC, Extended Mealy Machines, Mealy/Moore automata and recursive functions.

The paper [7] presents the results of research in which finite automata with output are used to construct a public-key cryptosystem (FAPKC). It is shown that the design employs weakly reversible linear and nonlinear automata endowed with input-output memory and a finite delay τ. The public key is defined by an ordered sequence of automata, an initial state, and τ. The private key consists of inverse automata, which states serve as secret parameters. In this configuration, the automaton graph, initial state, and delay handles expand the key space. They allow asymmetric operations based on machine reversibility. However, early FAPKC constructions provided only partial surjectivity guarantees at practical sizes. They also demonstrated limited resistance to related-key attacks, meet-in-the-middle attacks, and quantum-assisted attacks.

The paper [8] presents the results of a study of DAFA, a finite-state machine cryptosystem augmented with DES. It derives 16 DES key substitutions and creates linear and nonlinear automata, as well as their initial states, from these substitutions. And uses the resulting machines for encryption/decryption. It is shown that the experiments using more automata and adding a bit of random padding to each block increase the randomness of the ciphertext. FA modules give a way to get confusion and diffusion using the key schedules. For example, a bridge between ENT/NIST style statistics and differential or linear bounds was not built. Have not done an assessment of the security gains that FA layers give compared to DES subkeys. Also see that need performance and latency analysis, in benchmarks compared to optimized AES class implementations.

Paper [9] presents the results of a study on FSAaCIT. FSAaCIT combines a key-state machine cipher with chunk-level hash indexing. The combination lets the system do secure cloud deduplication. The cipher treats encryption as lossless, mappings of automata. The key length grows with the number of states. For example, 16 → ≈128 bits; 64 → ≈512 bits. The authors claim that the throughput for text sizes of 32–512 KB is higher, than AES. When ran the test it is possible to see the system at, about 364.76 KB/s for encoding and about 383.44 KB/s for decoding. The system shows a speed than AES, which gives 229.91 KB/s for encoding and 242.32 KB/s for decoding. The system also beats DES. The ENT statistics show entropy χ^2 in ranges a mean value near 127.5 and a serial correlation close, to zero. The system adds HMAC→SHA-1 Tag and SHA-256 over ciphertext (Hcode) to provide proof-of-possession. Data integrity verification also runs in the private cloud controller. The threat model covers attacks, external attacks and counterfeit possession scenarios. The shortfall probably comes from three things. First, large-scale machine building and reversal cause a state space. Second, randomness measurements do not turn into guarantees against attackers. Third, doing a comparison, against optimized reference ciphers on the same input output and hardware is expensive. All of those hurdles have made the stability hard to get. All of those hurdles have also made the attack-aligned evaluation rare, in the work.

The paper [10] shows the research results on an authenticated-encryption construction that combines FA cryptography with the Encrypt-then-MAC approach. The construction first encrypts the plaintext *t*, with an automaton that produces the ciphertext *y*. The construction then computes the tag *t* with a function that runs on a Mealy automaton. The construction sends the pair (*y*, *t*). The design of the construction aims to resist forgery. The design also aims to resist chosen-message attacks. The automaton non-linear behavior and state changes can be used to keep data secret and safe. When to pair an automaton with a MAC, it is possible to make the channel stronger. The automaton mixes the data. It spreads the data without using lookup tables. The encrypt-then-MAC layout follows a way to protect the data. The automaton representation also shows how the keys can live in the states in the transition functions in the output functions and even in the wiring. There are still issues. The first issue is security. Provable security needs IND-CCA and SUF-CMA guarantees that go beyond tests and the usual EtM folklore. The second issue is invertibility guarantees. Invertibility guarantees must hold at state sizes. Invertibility guarantees require surjectivity and unique decodability, for the chosen FA class. The third issue is efficiency. Comparative efficiency compares the system to AEAD baselines such, as AES-GCM and ChaCha20-Poly1305, are under controlled hardware I/O. The fourth issue is keying discipline. Keying discipline requires the automaton parameters, states, transition tables, Boolean functions, to be generated and rotated securely.

Designers used the approach in key nonlinear automata. The early single key nonlinear automata include extended Mealy automata [11]. In the key nonlinear automata, the secret contains transition functions. The secret also contains external variables. The user also sets the number of rounds the block size and the key length. Hybrid Mealy/Moore-based symmetric schemes [12] integrate a matrix, a generating function graph and reversible automata. These lines usually do not give invertibility proofs at scale. These lines do not give tests on the data (e.g. images/video). These lines do not give comparisons, with the AEAD standards under conditions.

FAPKC [7] follows the line of automaton ciphers. FAPKC offers clear tips, for modern symmetric high-speed use or for protecting images and multimedia. That gap makes choose automata for the guaranteed reversibility and makes embed FA blocks in the AES class pipeline. The advantage is the difficulty of factoring automata [13]. FAPKC to FAPKC3 was later improved in 1997 [14]. To FAPKC4, in 1999 [15]. FAPKC4 added FA with pseudo memory. In practice adjusting the states, transitions and automaton delay (pseudo-memory) expands the space. It still supports reversible mappings. Key space grows. Unresolved issues remain about engineering recommendations, for high-performance deployments and protecting multimedia. FAPKC-style proposals give recommendations for proving surjectivity and reversibility on a practical scale. FAPKC-style proposals also lack an approach to reconcile the estimates with attack models or to ensure real-time performance, on modern hardware. Several factors can be cited as explanations. Notice Challenges were noticed with the composition and decomposition of automata. The composition and decomposition of automata make the state space grow huge. The composition and decomposition of automata also make factoring composites hard. Benchmarking costs a lot compared to optimized AES-class baselines. Benchmarking costs a lot when compare benchmarking to optimized AES-class baselines and that makes benchmarking to run

tests. The statistical test does not match the proof frameworks. The proof frameworks are built for attacker models. The statistical test batteries are not built for attacker models.

Paper [16] proposes a symmetric key cryptosystem built on a sequential Mealy state machine. The automaton works as the encryption key. The automaton works as the decryption key when the automaton is used in reverse. The encryption process follows the machine $M = (Q, I, O, \delta, \lambda, q_0)$. First the plaintext is permuted by the function (P). Then the machine M changes the text into ciphertext. The inverse machine M' and the inverse function g^{-1} bring back the plaintext. The scheme looks for machines that do not lose information. That are weakly invertible. The scheme wants those machines to guarantee decodability. The scheme also wants the inverse machine M' to be hard for an attacker to build. The key length grows with the number of states ($|Q| \times 8$ bits; eight states → 64-bit key). The key length gets bigger when the number of states gets bigger. The key length can be adjusted by making the number of states. Authors built a Python prototype that uses 64-bit blocks. The prototype runs faster, than DES for message sizes that tested. The authors claim the prototype cuts the time needed by 30% compared to DES. The security considerations include that an attacker would have to try a number of machine keys in a brute-force search. The authors also think about how hard the reverse machine would be to reverse engineer.

This study view that there is no reversible performance-competitive framework. The framework must synthesize a S-box from finite automata method (FAM) primitives. The framework must guarantee properties such, as 8×8 bijectivity, nonlinearity $\geq$112, differential homogeneity $\leq$4 algebraic degree $\geq$7, no linear structures, no fixed points and good boomerang metrics. At present there is no framework that shows advantage over baseline AES in an attacker-focused evaluation. An evaluation would include the analysis, the linear analysis, the boomerang analysis and the related-key analysis. The research gap is clear. It needs a FAM-based algorithm that builds its S-box and places the S-box inside the AES-class SPN. The algorithm must improve the AES properties. The algorithm must give the reversibility guarantees. The algorithm must also give security proofs for each attack. The algorithm must also reach parity or a performance advantage in a comparison.

3. The aim and objectives of the study

The aim of the study is to improve the security of the FAM-based Advanced Encryption Standard algorithm by using its substitution block a lookup table called S-box. The study strengthens security, for multimedia data. It gives a template for future AES-class designs that use finite automata.

To achieve this aim, the following objectives were completed:

– to characterize the correlation structure before/after encryption by computing 2D and 3D correlation coefficients, including per-channel (R, G, B) correlations, for the original image and images encrypted by AES and the proposed method;

– to quantify intensity distributions via RGB-split and merged histograms and summary statistics (arithmetic mean, entropy), and to apply Chi-square goodness-of-fit tests for the original, AES-encrypted, and proposed-method images;

– to assess randomness and diffusion/confusion properties using the NIST SP 800-22 Statistical Test Suite together with NPCR and UACI analyses;

– to evaluate robustness to data loss by simulating block-based occlusion attacks and measuring the impact on decryption/recovery;

– to compare the proposed method with recognized approaches (e.g. AES, DES and recent alternatives) on all the above metrics.

4. Materials and methods

4. 1. The object and hypothesis of the study

The object of the study is the AES-128 – based image-encryption scheme.

The central hypothesis is that if replace the ShiftRows/MixColumns in AES-128 with an automaton and if pair the surjective finite automaton with a criteria-optimized 8-bit S-box in a 10-round CBC pipeline then will improve confusion and diffusion on images. This change will make the encryption spread the image data better. It is expected that the histograms are near-uniform. It is also expected that the inter-pixel correlations are negligible. The PSNR stays stable under noise or occlusion. The result should outperform AES on the metrics.

The study makes the following assumptions:

1. Replace AES ShiftRows and MixColumns with a finite automaton because expect the surjective finite automaton to preserve information-losslessness (decodability) at the machine level and to strengthen permutation and diffusion at the bit level. Let's define the FA and use the FA as a substitute for the AES ShiftRows and MixColumns layers in a 10-round AES-128 pipeline.

2. The security gains come from the custom 8-bit S-box that is made to meet limits. The custom 8-bit S-box replaces the Rijndael S-box in the SubBytes step of the pipeline.

3. The CBC mode gives security when the IVs are pseudorandom and the IVs do not repeat. In the description, the IV is a pseudorandom 128-bit value. One test variant builds the IV by computing an MD5 hash of the encrypted image and then treating the MD5 hash as a pseudorandom IV.

4. The statistical metrics and the differential metrics such, as NIST SP 800-22 NPCR, UACI and the robustness tests such, as occlusion serve as substitutes for more confusion and a smaller attack surface. It is possible to use the metrics and the differential metrics and the robustness tests when compare this work to the AES and DES baselines.

5. It is possible to notice that the FA is built as a mapping that covers the set of states. The FA makes each state go to a state and the FA does not lose any information. The FA works at the bit level. The FA copies the row shifts and the FA copies the column mixing. The FA is meant to give diffusion, then the ShiftRows/MixColumns pair.

Here are the simplifications used in the study:

1. Testing a pipeline that uses AES-128. Fix the structure to ten rounds. Use the standard AES-128 key schedule. Change the diffusion layer (ShiftRows/MixColumns → FA). Change the substitution layer (custom S-box). Leave all AES-128 mechanics. All other AES-128 parts stay the same.

2. Evaluate security, in CBC mode with the random or pseudorandom 128-bit IV (including a variant that uses the IV). Do not use the AEAD wrapping and modes (e.g. CTR, GCM).

3. Inputs treat as 8-bit RGB. Inputs process in 128-bit blocks. TIFF for I/O use with the inputs. The simple pre-permutation breaks the locality of the inputs.

4. 2. Materials

The method was tested on benchmark images from the available USC–SIPI Image Database. The USC–SIPI Image Database is a testbed for image processing and encryption studies. Two categories of images were selected. Two categories of images were selected based on their resolution and their content diversity.

The first category has four images. Each image is 256 × 256 pixels:

– "4.1.04 (Female)" – a standard portrait image frequently used for visual quality analysis;

– "4.1.05 (House)" – an architectural image with structured edges and textures;

– "4.1.06 (Tree)" – a natural image exhibiting fine spatial details;

– "4.1.08 (Jelly Beans)" – a composition with varied color and texture distribution.

The second category consists of four high-resolution images with dimensions of 512 × 512 pixels:

– "4.2.03 (Mandrill)" – a highly detailed animal image with complex textures;

– "4.2.05 (Airplane)" – a structured image with distinct contours and regions;

– "4.2.06 (Sailboat on Lake)" – a natural scene combining water, sky, and manmade objects;

– "4.2.07 (Peppers)" – a popular test image for evaluating color preservation and encryption fidelity.

It is possible to build the test set from the USC–SIPI database. The test set includes a mix of textures, content and spatial layouts. The variety helped to probe the cipher under conditions. Robustness, statistical security and perceptual distortion were measured on both scenes and unstructured scenes. Its suite is representative. It avoids duplication and excludes overlapping candidates.

4. 3. Custom S-box

In this paper, a software tool for analyzing and synthesizing cryptographic S-boxes written in Python3, described in [17], is used to generate new optimal S-boxes. The tool provides the user with the ability to analyze already known cryptographic S-boxes. New S-boxes are synthesized based on flexibly configurable and user-defined values of the optimality criteria. The tool allows generating 8-bit bijective S-boxes using a generalized algebraic method for generating Rijndal-like S-boxes [18]. Following this approach, it is possible to create new S-boxes with cryptographic properties identical to or even superior to the S-boxes used in AES. This software tool can also be used to generate optimal 8-bit S-boxes using a combined method: a generalized algebraic method [18] + a gradient descent method [19].

When generating the S-box, the combined method [17] builds the S-box:

1) Generating an S-box using a generalized algebraic method.

2) Randomly swap NP values of the S-box. Generate a S-box. It is possible to take the NP parameter that the user specifies. It is possible to take the set of criteria values that the user specifies. For example, the recommended set is: uniformity – no, than 8, nonlinearity – no less than 100 algebraic degree – no less, than 7, algebraic immunity – maximum value (3, 441) absence of fixed points and opposite fixed points.

3) The obtained S-box is checked step by step, against the criteria. The S-box check considers the complexity of each criterion. If any criterion is not met the process returns to step 2.

4) Completion. The result is a S-box that meets the given criteria. The S-box meets the given criteria.

When look at Fig. 1 it is possible to see the substitution box (S-box) used in the AES encryption algorithm in this study. The S-box is a 16 × 16 matrix that holds 256 8-bit values. Each entry maps a byte input, to an output byte. The S-box replaces the Rijndael S-box used in AES. The S-box uses a transformation that comes from field arithmetic over GF(2^8).

F8	F9	4A	24	A1	6F	96	0D	66	AB	01	B9	CF	53	30	56
B7	41	63	03	36	DF	6A	42	51	88	1F	C9	9C	C7	AF	32
6D	BE	16	BA	07	81	37	ED	9F	A3	59	0B	B1	34	A5	65
1E	E3	C0	4C	39	72	52	F4	CA	4D	55	2E	61	21	9D	E5
00	F3	DB	17	8F	92	D9	6C	35	D4	76	48	2D	69	40	E8
79	85	67	F1	1A	94	33	E6	6E	FC	9E	D1	64	D6	04	5D
8B	11	47	19	E4	C6	A2	D0	2A	28	BD	8E	AD	1B	FE	0C
E1	50	10	98	1C	0E	93	BC	06	DD	26	12	78	A9	44	08
84	A8	4F	5F	5B	75	3D	14	71	70	CD	38	5A	7D	B2	49
2C	B5	EE	43	BF	D8	A0	FD	20	C4	02	EA	A4	D7	F0	AA
0A	D2	74	7C	05	A7	4E	7B	89	E0	CE	F5	2F	C2	F7	31
B3	77	FA	25	CB	C1	5E	7A	B6	E9	EF	6B	86	09	18	9A
73	CC	3E	22	15	7E	3A	23	F6	57	E7	AE	D5	B0	EC	DE
91	2B	90	29	68	B4	C3	54	60	C5	3B	3F	FB	4B	82	13
46	9B	AC	95	8C	0F	C8	E2	8A	99	83	27	7F	3C	DA	BB
87	45	58	D3	97	FF	8D	1D	B8	F2	62	EB	A6	5C	80	DC

Fig. 1. Custom S-box for modified AES algorithm

The custom S-box enhances nonlinearity and reduces correlation. The custom S-box provides security and strong linear security. When connect the custom S-box to an AES pipeline add another substitution step. This addition makes the encryption scheme more resistant, to attacks from methods and new methods. These substitution components are key to achieving obfuscation. Obfuscation is one of Shannons ideas for designing ciphers. Ensuring the strength of the encryption scheme is important. The pseudocode creates the constants for the S-boxes. The constants follow the steps shown in the pseudocode:

```
FUNCTION BuildSBoxConstants(mode):
    IF mode == "AES":
        affineConst ← 0x63
        polynomial  ← 0x11B
        affineMatrix[8] ← [
            10001111b,
            11000111b,
            11100011b,
            11110001b,
            11111000b,
            01111100b,
            00111110b,
            00011111b
        ]
    ELSE IF mode == "RANDOM":
        affineConst ← RandomByte()        // 0..255
        polynomial  ← RandomChoice([11B,12D,
14D,163,165,169,171,187]h)
        seed ← RandomByte()
        FOR i IN 0..7:
            affineMatrix[i] ← RotateByte
(seed, i)     // cyclic shift by i
    RETURN (affineConst, affineMatrix, polynomial).
```

4. 4. Ensemble algorithm based on FAM using S-Box

The modified AES-128 design replaces ShiftRows and MixColumns with a surjective finite automaton to increase permutation complexity. It also uses a custom S-box derived from a chosen irreducible polynomial and an affine transformation matrix. Encryption runs for 10 rounds in CBC mode. This method is based on the custom S-box cipher, and allows for variable key and block sizes. The number of rounds in custom S-box was defined as

$$\text{Rounds} = \max(\text{block length}, \text{key length})/32 + 6,$$

where Key length = 128 bits, Block size = 128 bits. So

$$\text{Rounds}=\max(128,128)/32 + 6 = 4 + 6 = 10.$$

A finite automaton $A = <Q,X,Y,\delta,\lambda>$ is called surjective if for any output sequence $y \in Y^*$ there exists an input sequence $x \in X^*$ such that, for the initial state $q \in Q^*$, the automaton A, when reading the sequence x, produces the output sequence y. It's means that for any $y \in Y^*$ there exists $x \in X^*$ such that: $\lambda^*(q_0, x) = y$, where $\lambda^*(q_0, x)$ is the output sequence of the automaton for the initial state q_0 and the input sequence x [4]. In other words, surjective automata are finite automata that satisfy the conditions of surjectivity, i.e. it is necessary and sufficient that in each row of the output table each output symbol occurs at least once. The encryption algorithm is shown in Fig. 2. The encryption algorithm is examined. It is possible to see that the encryption algorithm uses a custom AES-128 modification. The custom AES-128 modification adds an automaton and a custom S-box.

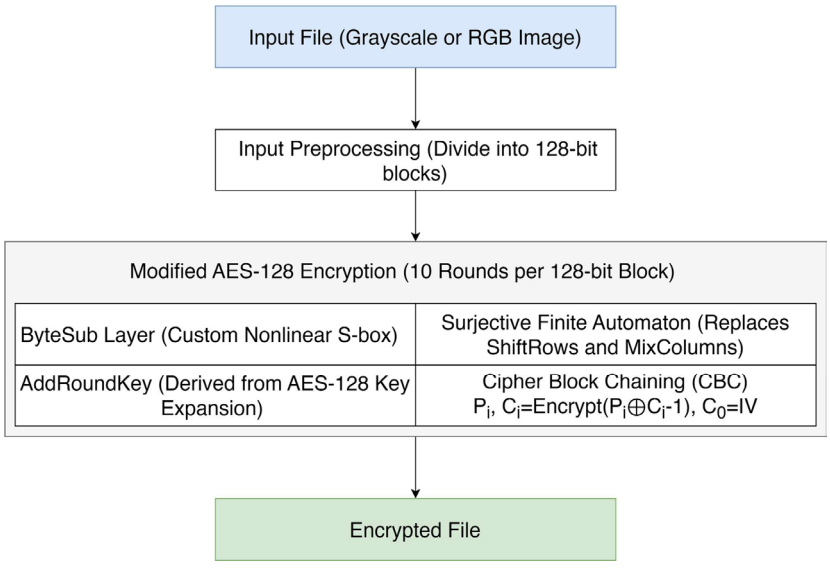


Fig. 2. Graphical flow chart outlining this encryption process

First step is Input Preprocessing. It is possible to take a Grayscale or RGB image as the input. The image is split into 128-bit blocks, which are 16 bytes each because AES requires that size. It is also possible to rearrange the image data to hide locality before the image enters the AES pipeline.

The second step uses an AES encryption pipeline. The pipeline works on each 128-bit block. The block goes through ten rounds of the following transformations:

1. ByteSub (SubBytes) – the substitution layer. ByteSub (SubBytes) replaces each byte of the state matrix with a value, from the given nonlinear S-box. The S-box is built as follows:

1. 1. Input inversion, in GF(2^8) using polynomial – x^8 + + x^6 + x^5 + x^2 + 1. The system uses the eight-bit identifier and the additive constant, for the affine postprocessing step. The affine matrices contain eight-bit rows. The affine matrices are the identifiers of the selected polynomials. The irreducible polynomials define the field GF(2^8) for the inversion step. The hexadecimal values are the affine constants. The set of parameters, for the configuration is a field identified by the identifier "10000000". The set of parameters does not specify the polynomial. The set of parameters also includes an affine c = 0xF8. The set of parameters also includes an affine matrix A of size 8 × 8. The input domain is all bytes x∈{0, … 255}. The construction applies the inverse field $y=x^{-1}$ in GF(2^8). The construction assigns $y = 0$ for $x = 0$ unless the construction says otherwise. The construction then uses the affine map $(x) = Ay \oplus c$. The construction gives pseudocode for obtaining the inverse element, in GF(2^8) modulo a polynomial:

```
FUNCTION GFInverse(a, poly):
    IF a == 0: RETURN 0
    x ← a
    y ← poly
    u ← 1
    v ← 0
    WHILE x > 1:
        j ← bit_length(x) − bit_length(y)
        IF j < 0:
            SWAP(x, y)
            SWAP(u, v)
            j ← −j
        x ← x XOR (y << j)
        u ← u XOR (v << j)
    RETURN u AND 0xFF.
```

1. 2. The affine step works over the binary field $GF(2)$. So, every operation is *XOR* and every matrix entry is 0 or 1. It is possible to take the byte produced after inversion and view it as an 8-bit column vector. The step applies an 8 × 8 binary matrix and then *XORs* a fixed 8-bit offset. In the figure, the matrix shown is the identity, so the matrix part leaves the bits unchanged. The only effect is the *XOR* with the constant vector. Let's affine transformation with matrix:

$$\begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{matrix}. \quad (1)$$

1. 3. Affine vector is f8. The vector f8 is a hexadecimal constant equal to 0xF8. In binary this is 11111000, meaning the top five bits are 1 and the bottom three bits are 0. The affine step flips (*XORs*) the top five bits of the input byte

and leaves the bottom three bits unchanged. Pseudocode for an 8×8 affine transformation is in GF(2):

```
FUNCTION AffineTransform (byte, affineMatrix [8], affineConst):
    result ← 0
    FOR i IN 0..7:
        bit ← 0
        FOR j IN 0..7:
            a ← (byte >> j) AND 1
            b ← (affineMatrix[i] >> (7 − j)) AND 1
            bit ← bit XOR (a AND b)
        result ← result OR (bit << i)
    RETURN result XOR affineConst.
```

2. Surjective Finite Automaton (Substitute, for ShiftRows + + MixColumns). The state is an 8 × 16 array of bits. The input sequence is called *x**. It is possible to build the states of the finite automaton at random. The surjective finite automaton maps each state to a state so there is no loss of information. The surjective finite automaton simulates both row shifts and column mixing. The surjective finite automaton works on bit-level data to shuffle and spread the state bits securely than transformations.

3. AddRoundKey – Key Mixing. It is possible to notice that the round key from the AES-128 expansion does not XOR the state. The 128-bit key expands into ten keys, for compatibility.

Third step. Cipher Block Chaining (CBC) Mode Implementation. Initialization Vector (IV) is random or pseudorandom 128-bit block. For each plaintext block

$$P_i,\ C_i = Encrypt(P_i \oplus C_i - 1),\ C_0 = IV. \quad (2)$$

In testing use TIFF images. The original, encrypted, and decrypted images are stored in this format.

Block processing order:

1. Encryption:

– Round 0: addRoundKey;

– Rounds 1–9: subBytes → shiftRows → toAutomata → addRoundKey;

– Round 10: subBytes → shiftRows → addRoundKey.

2. Decryption:

– Round 0: addRoundKey;

– Rounds 1–9: invShiftRows → invSubBytes → addRoundKey → toInvAutomata;

– Round 10: invShiftRows -> invSubBytes -> addRoundKey.

The MD5 hash of the encrypted image was used as the initialization vector (IV) for AES tests modified with finite automata.

4. 5. Two-dimensional correlation coefficient

It is possible to look at pixel-level dependency and statistical redundancy using the two- correlation coefficient (2D-CC) [20]. The two-dimensional correlation coefficient shows relationships, between pixels in the horizontal and vertical directions. The 2D-CC method starts with the selection of pixel pairs. For each image original and encrypted extract pixel pairs in the vertical and diagonal directions. The adjacent pixel pairs represent relationships, in the domain of the image. It is possible to plot the intensity values of pixel pairs as scatter plots to see the distribution of dependencies. When most points sit on the diagonal the correlation is strong. When the points are spread evenly the data look random. It is possible to compute the correlation coefficient *r* for each direction with the formula that uses *x* and *y* as the grayscale values of pixels and *N*, as the number of sampled pixel pairs. For images expected values of $r \approx 0.9$ or above

$$r = \frac{N\sum xy - \sum x \sum y}{\sqrt{\left(N\sum x^2 - \left(\sum x\right)^2\right)\left(N\sum y^2 - \left(\sum y\right)^2\right)}}. \quad (3)$$

An ideal cipher gives correlation coefficients to zero. Those, near zero values show independence between neighboring pixels. That independence means diffusion. The method will be applied to the channel, the channel and the blue channel separately. The method will use the test images “4.1.04 Female”, “4.1.05 House” and “4.2.03 Mandrill”. For each case the method will generate visual scatter plots. The visual scatter plots will compare the image correlations, with the encrypted image correlations. The original image correlations will be shown side by side with the encrypted image correlations.

4. 6. Three-dimensional correlation coefficient

Independence and cipher strength evaluated with the three-dimensional correlation coefficient (3D-CC) [21]. The three-dimensional correlation coefficient extends the two-correlation by adding an axis that can be spatial or chromatic. The three-dimensional correlation coefficient lets view the data, in three dimensions and measure the dependencies. The input preparation treats each RGB image as a three-structure, with axes *x* (horizontal), *y* (vertical) and *z* (color or layer depth). The analysis runs per channel. The analysis runs per channel for the R channel, the G channel and the B channel. In the 3D scatter plots, the x axis and the y axis show the neighboring pixel intensities and the *z* axis shows the position layer or the spatial depth. The unencrypted images usually give stacked layers or planar bands. The stacked layers or planar bands show correlation. The encrypted images give a uniform point cloud. The uniform point cloud shows dependence. The compact and aligned geometries show correlation. The near-uniform dispersion shows correlation. Repeat the method across USC-SIPI images. Each channel was plotted individually. It is possible to observe how the encryption breaks the pattern and the color pattern.

4. 7. Histogram analysis

Histogram analysis is a method [22] for check the security and statistical uniformity of encrypted images. In image cryptography, histogram analysis works as a tool to see if the encryption algorithm removes patterns in pixel intensity distribution. For each image, the intensity histogram was computed in two ways: RGB histograms and a combined RGB histogram. Histograms madden, for the red channel, the channel and the blue channel and made a combined histogram to see the overall distribution. The peaks and troughs tell about the brightness levels and the natural shape of the scene. A flat histogram tells that the histogram shows all brightness levels equally. If the histogram shows all intensities equally the statistical redundancy disappears. When it is possible to see non-uniform shapes in the histogram it is known that there is correlation between neighboring pixels. When it is possible to see a near-uniform profile in the histogram it is known that the image has been randomized well. A near-uniform profile also tells that the ciphertext is visually and statistically uninformative.

4. 8. Entropy, Chi-square statistical test, arithmetic mean of pixel intensities analysis of images

Entropy analysis is a way to check the randomness in images [23]. It is possible to look at the entropy value when work with image processing. A high entropy value tells that the image data is random in a sense. Entropy analysis helps to fight entropy-based attacks and keep encryption strong. The entropy H of an image uses Shannons entropy formula

$$H=-\sum_{i=0}^{255}P(i)\log_2 P(i), \tag{4}$$

where $P(i)$ – the probability that the pixel intensity value i occurs. When look at an 8-bit image (pixel range [0, 255]) expect the ideal entropy value to be 8 bits. The ideal entropy value tells that there is randomness and uniform distribution.

The Chi-square statistical test does not assume a shape of the data. The Chi-square statistical test checks the uniformity of pixel intensity distributions, in encrypted images [24]. In image cryptography the Chi-square statistical test measures the deviation of the histogram of an encrypted image from a uniform distribution. The Chi-square statistical test is a check for resistance to attacks. The Chi-square value $\chi 2$ is calculated as

$$\chi 2=\sum_{i=0}^{255}\frac{\left(O_i-E\right)^2}{E}, \tag{5}$$

where O_i – the observed frequency of intensity level i. E – the expected frequency. For a distribution the expected frequency is $E=N/256$, where N's – the total number of pixels, in the channel. The Chi-square statistic is close to 255 which is the degrees of freedom for an 8-bit histogram. The Chi-square statistic close to 255 tells that the encrypted image's histogram is nearly uniform. It is also possible to see that a large drop in the chi-square statistic from the image, to the encrypted image shows that the algorithm removes patterns and the algorithm improves visual security. The mean of pixel intensities is a useful measure of distribution symmetry and brightness balance [25]. In encryption studies the arithmetic mean helps to see whether pixel values have been sufficiently randomized to remove structure or bias. The arithmetic mean μ is calculated as

$$\mu=\frac{1}{N}\sum_{i=1}^{N}P_i,$$

where P_i – the pixel intensity, N – the number of pixels, in the image or color channel.

4. 9. Statistical test suite

The randomness of the sequences is assessed with NIST's Statistical Test Suite (STS). The suite is a set of tests, from the National Institute of Standards and Technology [26]. The NIST STS processes the bitstreams that come from the encrypted images. To form the bitstreams the process extracts the bits of the pixel values or uses the full 8-bit values of each RGB channel. The study will use a subset of tests, from the NIST STS including:

– frequency (Monobit) Test checks if the number of zeros and ones are the same. Use the frequency (Monobit) Test to count the zeros and the ones and compare them.

– block frequency test – checks the evenness of the 0 s and the 1s, in each M-bit block;

– run the test – this test checks whether the oscillation, between 0 s and 1s is too fast or too slow;

– longest run of ones – measures the maximum sequence of 1s in fixed-size blocks;

– Binary Matrix Rank Test – checks dependency among the fixed size matrix blocks. Binary Matrix Rank Test looks at how the matrix blocks depend on each other;

– Spectral (DFT) test – detects repetitive patterns using Discrete Fourier Transform;

– Overlapping Template Matching Test – use the non-overlapping Template Matching Test to count how many times a specific bit pattern appears;

– Overlapping Template Matching Test – similar to above, but allows overlapping patterns;

– Universal Statistical Test – measures how compressible the sequence's;

– Linear Complexity Test –use linear complexity Test to check the complexity of the sequence. Linear Complexity Test shows if the sequence is simple or not;

– Serial Test – checks the uniformity of overlapping m-bit patterns. It looks at each overlapping m-bit block;

– Approximate Entropy Test – compares how often overlapping blocks of two lengths appear. The approximate entropy test looks at how each pattern shows up when the blocks overlap. The approximate entropy Test works well;

– Sums (Cusum) Test – checks how much the data differs from a walk. The cumulative Sums (Cusum) Test reveals the size of the deviation;

– Random Excursions Test – count how times the random walk visits each state. The Random Excursions Test reports the number of visits, to each state;

– Random Excursions Variant Test – a refined version of the excursion test focusing on specific states.

All tests were checked using the significance level α set to 0.01. It was possible to call a test of a passing when the p-value is larger than 0.01. The p-value larger than 0.01 means the null hypothesis of randomness cannot be rejected.

4. 10. Number of pixels change rate and unified average changing intensity

To measure the sensitivity and the differential security of the proposed image encryption algorithm used two metrics: Number of Pixels Change Rate (NPCR) [27]. Unified Average Changing Intensity (UACI) [28]. Number of Pixels Change Rate (NPCR) and Unified Average Changing Intensity (UACI) tell how well the encryption scheme can resist attacks. Differential attacks change a part of the input. Try to find patterns in the output. The metrics let's see if the encryption scheme can stop those attacks.

When compare two encrypted images $C_1(i, j)$ and $C_2(i, j)$ check each pixel. Set D to 1 if $C_1(i, j)$ and $C_2(i, j)$ are different. Set D to 0 if they are the same. The width of the image is W and the height of the image is H. The NPCR is then defined as:

$$NPCR=\frac{1}{W\times H}\sum_{i=1}^{W}\sum_{j=1}^{H}D(i,j)\times 100\%, \tag{6}$$

$$D(i,j)=\begin{cases}1, \text{if } C_1(i,j)\neq C_2(i,j),\\ 0, \text{if } C_1(i,j)=C_2(i,j).\end{cases} \tag{7}$$

The ideal NPCR value, for a 256-level grayscale image should be to 99.6%. The NPCR value shows that all the pixels are affected by a one-pixel change, in the image.

UACI measures the intensity of differences, between two images. UACI tells how much the pixel values change

because of a change in the input. UACI matters as a test, for how unpredictable the encryptions. UACI is defined as

$$UACI = \frac{1}{W \times H \times 255} \sum_{i=1}^{W} \sum_{j=1}^{H} \left| C_1(i,j) - C_2(i,j) \right| \times 100\%. \quad (8)$$

In these tests a UACI value 33% works best for 8-bit images. The UACI value shows diffusion in the encryption algorithm.

4. 11. Peak signal-to-noise ratio

Use peak signal-to-noise ratio (PSNR) to measure how well decryption keeps image quality when the data is degraded. It is possible to check the PSNR after each test. Noise or occlusion attacks can degrade the data. A higher PSNR means the recovered image is closer, to the original. The PSNR gives a gauge of the encryption-decryption pipeline robustness and reliability. Compute PSNR using the MSE – Mean Squared Error between the image I and the decrypted image K. It is possible to use W for the width H, for the height of the image. Use MAX for the pixel value of the images. It is express PSNR in decibels:

$$MSE = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} \left[I(i,j) - K(i,j) \right]^2, \quad (9)$$

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX^2}{MSE} \right), \quad (10)$$

where W and H – the width and the height of the image. MAX – the pixel value that the image can have. For an 8-bit grayscale or color image, the maximum pixel value is 255. The PSNR value rises when the visual quality improves and the distortion drops. A PSNR value of 30 dB or higher is usually called acceptable because the encrypted image and the decrypted image look the same to the eye. A PSNR value of 40 dB or higher means the reconstruction is very close to the original. PSNR analysis gives a view of the error stability of the cryptosystem when add distortions, like noise injection and when face accidental data loss like occlusion. PSNR analysis shows high PSNR stays stable across images and attacks. High PSNR across images and attacks supports the strength and stability of the proposed image encryption method.

5. Results of the image encryption

5. 1. Pre/post encryption correlation structure by calculating two-dimensional and three-dimensional correlation coefficients

Fig. 3 shows the two-dimensional correlation analysis of pixels, in the image "4.2.03 Mandrill" and its encrypted forms. Look at each subfigure. It is possible to see scatter plots. The scatter plots show the correlation between pixel intensities in the direction (top plot) the vertical direction (middle plot) and the diagonal direction (bottom plot). The horizontal plot is at the top the vertical plot is in the middle and the diagonal plot is, at the bottom.

The method reduces the correlation between pixels. The method reduces correlation in horizontal and vertical directions. The method matters for images that have textures. A strong encryption method should break the links so that statistical attacks cannot use the links. The results demonstrate the method consistently on reference images. Textured cases such as Mandrill and Peppers show large drops in correlation. The algorithm shows that the method works on textures.

Fig. 4 shows the 3D scatter surface plots. The 3D scatter surface plots illustrate the correlation of pixel intensities, in the encrypted image "4.2.03 Mandrill". The set of plots displays the pixel value relationships across the three directions – horizontal, vertical and diagonal.

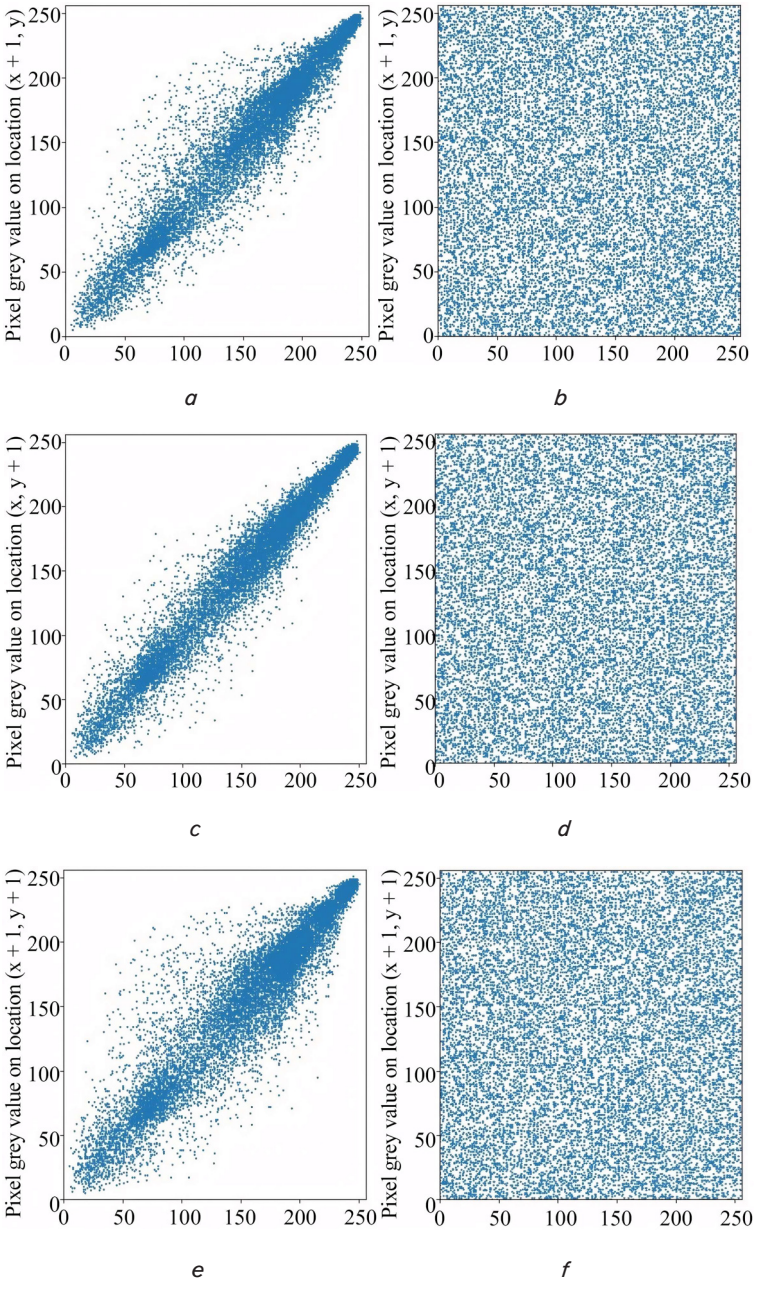


Fig. 3. Two-dimensional correlation coefficients of adjacent pixels in image "4.2.03 (mandrill)" before and after encryption: *a* – original horizontal direction; *b* – encrypted horizontal direction; *c* – original vertical direction; *d* – encrypted vertical direction; *e* – original diagonal direction; *f* – encrypted diagonal direction

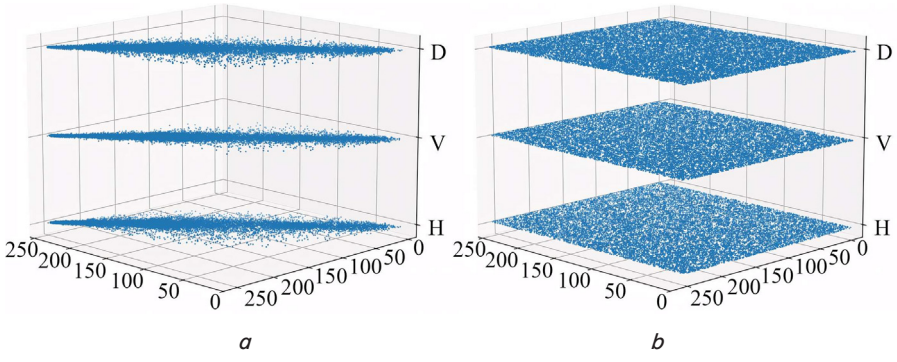


Fig. 4. Three-dimensional correlation surface plots of encrypted image "4.2.03 (mandrill)": *a* – original HVD directions; *b* – encrypted HVD directions

3D surfaces with ridges and peaks that show local connections because of the texture and the continuity of the structure. The encrypted images shown in this figure surfaces that look random with small changes, in height. The encryption process breaks the built-in connections between pixels. The drop in correlation across all images shows that the encryption algorithm breaks the correlation, between the pixels. When the decorrelation works the ciphertext does not keep any pattern or shape marks of the image. Statistical attacks become much harder. Differential attacks become much harder. Correlation maps appear uniform, for any scene. Correlation maps show the pattern no the scene. Uniform correlation maps indicate performance. Stable performance does not depend on image complexity. Image complexity does not change performance. Visual checks confirm a reduction, in pixel correlation. Pixel correlation drops in every direction. Pixel correlation drops across all color channels.

Table 1 shows pixel intensity correlation coefficients for the three-color channels – Red, Green and Blue. Table 1 compares pixel intensity correlation coefficients for the encrypted versions of eight images: "4.1.04 (Female)", "4.1.05 (House)", "4.1.06 (Tree)", "4.1.08 (Jelly beans)", "4.2.03 (Mandrill)", "4.2.05 (Airplane)", "4.2.06 (Sailboat on Lake)", and "4.2.07 (Peppers)". Calculate the correlation coefficients, in the horizontal (HC) vertical (VC) and diagonal (DC) directions. The calculation shows how strong the spatial relationships are between pixels.

In the images the correlation coefficients, in every direction and in every color channel are close to one above 0.9. The correlation coefficients indicate similarity because natural images have a structured form. After encryption the correlation coefficients fall to near zero between -0.03 and 0.03. The result confirms that encryption breaks similarity and removes repeating patterns from the pixel data. The positive correlation is high in the images. After the encryption algorithm processes the images the positive correlation becomes zero. The drop in the correlation shows the encryption algorithm removes the dependencies. The encryption algorithm is strong. The encryption algorithm protects the encrypted images from the analysis and the cryptanalytic attacks.

5. 2. Quantitative assessment of intensity distribution using RGB split-merge histograms and summary statistics

Fig. 5 presents a comparison of histograms for the image and the encrypted image of the color images. The top row shows the RGB separated histograms for the image. The top row includes a red channel histogram, a green channel histogram, a blue channel histogram and a combined histogram for the image. The bottom row shows the set of histograms for the encrypted image. The bottom row includes a red channel histogram, a green channel histogram, a blue channel histogram and a combined histogram for the encrypted image.

The original images have histograms. In the images, the RGB histograms show peaks and valleys and the histograms reflect a not even spread of pixel intensities. The histogram patterns match the shapes, surface patterns and color spread in the image and the histogram patterns can be used for math attacks or image rebuilding. The encrypted image looks different. In contrast, the encrypted image shows even histograms in all three-color channels.

Table 1

Horizontal, vertical, and diagonal correlation coefficients of RGB channels for original and encrypted images

Images/Channels	Red-HC	Red-VC	Red-DC	Green-HC	Green-VC	Green-DC	Blue-HC	Blue-VC	Blue-DC
4.1.04 Original	0.9800	0.9891	0.9737	0.9667	0.9710	0.9467	0.9555	0.9432	0.9261
4.1.04 Encrypted	0.0112	0.0037	–0.0134	0.0056	–0.0025	0.0096	0.0010	0.0065	0.0019
4.1.05 Original	0.9198	0.9517	0.8735	0.9089	0.9759	0.8849	0.9035	0.9747	0.8827
4.1.05 Encrypted	0.0140	–0.0019	–0.0059	–0.0034	–0.0241	–0.0091	–0.0086	0.0262	0.0062
4.1.06 Original	0.9150	0.9405	0.8735	0.9385	0.9448	0.8981	0.9315	0.9312	0.8911
4.1.06 Encrypted	–0.0165	–0.0165	–0.0077	0.0070	0.0028	–0.0193	0.0049	0.0046	–0.007
4.1.08 Original	0.9676	0.9672	0.9394	0.9504	0.9607	0.9144	0.9379	0.9541	0.8974
4.1.08 Encrypted	–0.0119	–0.0061	0.0086	0.0122	–0.0211	0.0004	0.0127	–0.0078	0.0074
4.2.03 Original	0.9583	0.9724	0.9384	0.9536	0.9724	0.9259	0.9648	0.9803	0.9468
4.2.03 Encrypted	–0.0229	–0.0008	–0.0056	0.0006	–0.0048	–0.0071	–0.0212	0.0251	0.0121
4.2.05 Original	0.9396	0.9569	0.8956	0.9294	0.9463	0.8799	0.9135	0.9284	0.8624
4.2.05 Encrypted	–0.0199	–0.0012	0.0080	–0.0042	–0.0296	0.0044	–0.0042	0.0026	0.0039
4.2.06 Original	0.9270	0.9158	0.8918	0.9144	0.8860	0.8582	0.8651	0.8244	0.7549
4.2.06 Encrypted	–0.0073	0.0084	–0.0069	–0.0096	0.0051	–0.0004	0.0040	–0.0232	–0.025
4.2.07 Original	0.9185	0.8895	0.8906	0.9420	0.8967	0.8942	0.9416	0.9247	0.9049
4.2.07 Encrypted	–0.0192	0.0325	–0.0089	–0.0098	–0.0150	–0.0097	0.0005	–0.0108	0.0241

The evenness shows a spread of pixel values, with no patterns or strong frequencies and the evenness is an important sign that the encryption worked. It is possible to notice that the combined histogram of the encrypted image shows the behavior. The combined histogram of the encrypted image displays an even intensity distribution, across the range of 0–255. The transformation from the histogram in the image to the random histogram in the encrypted image shows how the encryption algorithm works. The result makes the encrypted image more resistant to histogram based and statistical attacks.

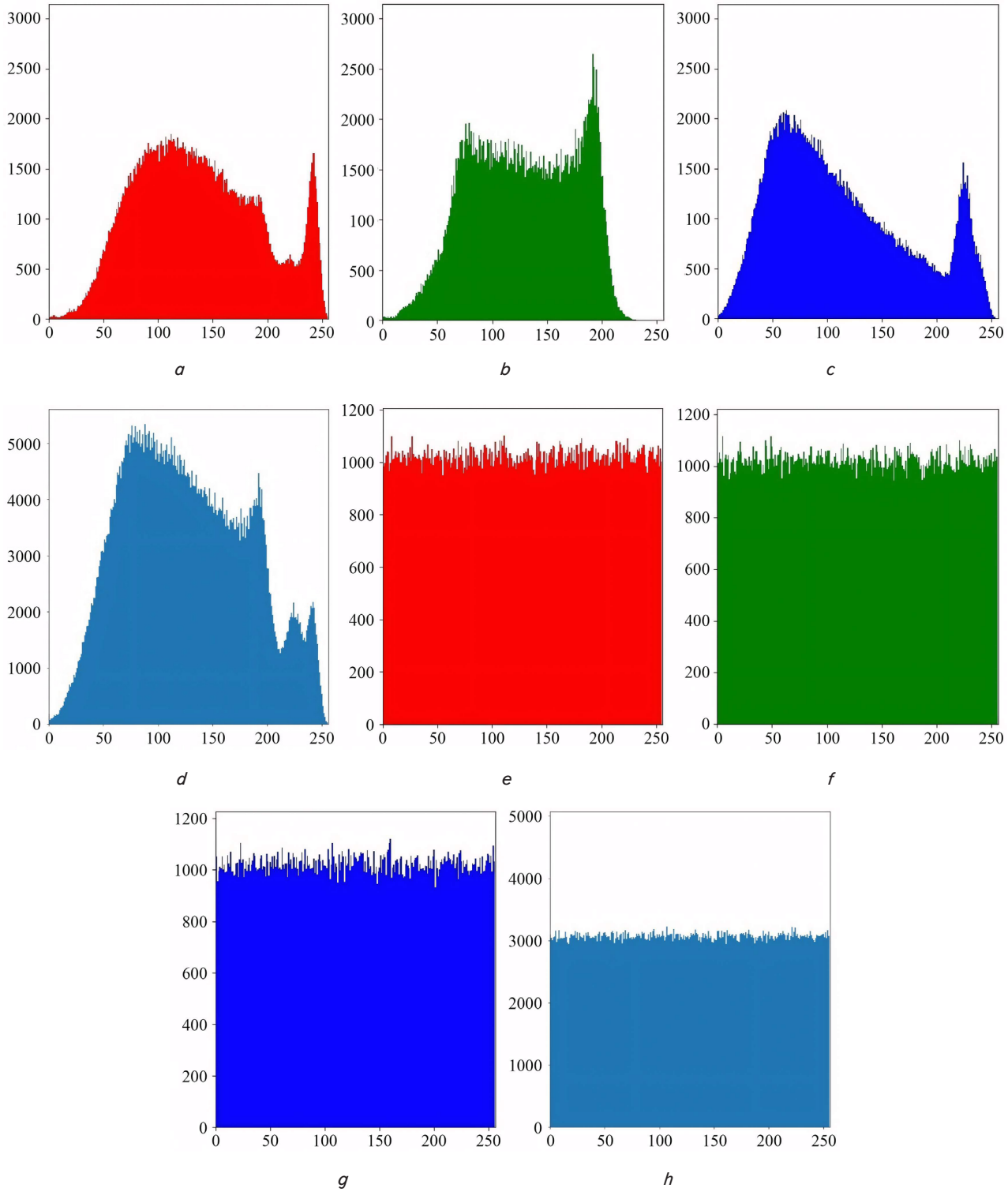


Fig. 5. Red, green, and blue and combined histograms of original and encrypted image “4.2.03 (mandrill)”:
a — original red channel; *b* — original green channel;
c — original blue channel; *d* — original combined channel;
e — encrypted red channel; *f* — encrypted green channel;
g — encrypted blue channel; *h* — encrypted combined channel

Fig. 6 shows a comparison of Shannon entropy values for the images and for the encrypted images. The comparison uses two encryption approaches: the Classic AES algorithm and the proposed encryption method. It is possible to notice that the evaluation includes RGB color images and their separate red, green and blue channels. Notice that the evaluation covers eight test images: "4.1.04 (Female)", "4.1.05 (House)", "4.1.06 (Tree)", "4.1.08 (Jelly beans)", "4.2.03 (Mandrill)", "4.2.05 (Airplane)", "4.2.06 (Sailboat on Lake)" and "4.2.07 (Peppers)".

The original images have entropy values below the maximum of 8. The entropy values vary across the channels. The variation shows statistical redundancy that natural images usually have. Natural images have pixel intensities that're not random. That are structured. After encryption, AES and the proposed method produce entropy values that're close, to the maximum of 8.0 for every channel and every image. The encrypted images have entropy values above 7.99. This shows that both methods make images where pixel intensities are spread out evenly. The high entropy means that little information leaks and that the cipher images resist attacks that look at entropy statistics. The proposed encryption method works as or a little better, than classic AES [29, 30] in cases for images that have low original entropy, such, as "4.1.05" and "4.1.08." These results show that the proposed encryption method keeps a level of encryption performance.

Table 2 shows the results of the Chi test. The Chi-square test looks at whether the pixel value distributions, in the original images and in the encrypted images. Applied two encryption techniques: Classic AES and the proposed method. The table compares the pixel value distributions for Classic AES and, for the proposed method. The test covered eight images: "4.1.04 (Female)", "4.1.05 (House)", "4.1.06 (Tree)", "4.1.08 (Jelly beans)", "4.2.03 (Mandrill)", "4.2.05 (Airplane)", "4.2.06 (Sailboat on Lake)" and "4.2.07 (Peppers)".

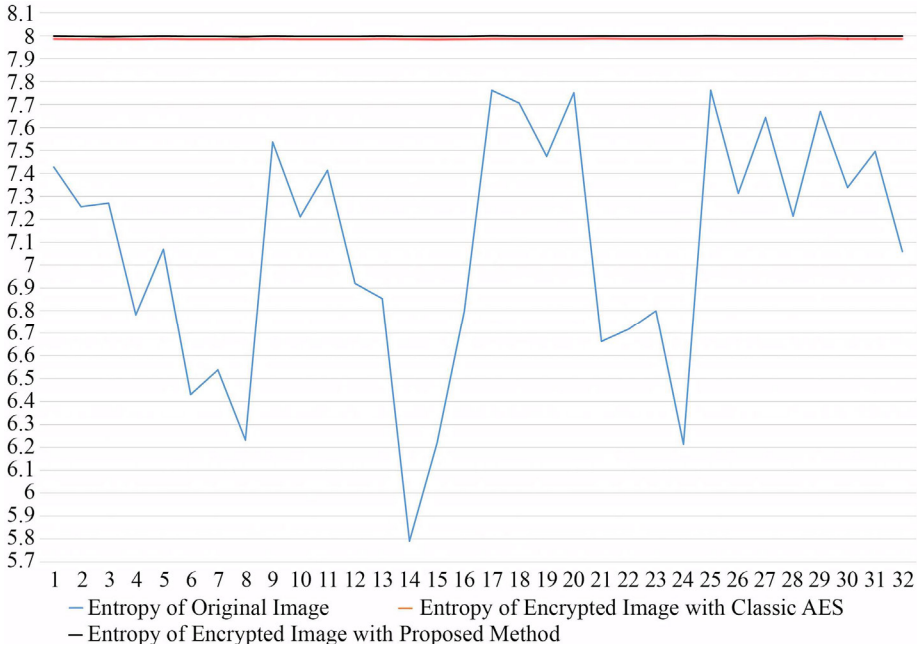


Fig. 6. Shannon entropy values of original and encrypted images using classic AES and the proposed encryption method

Table 2

Chi-Square statistical test results for original and encrypted images using classic AES and the proposed method

Images	Chi-square of original image	Chi-square of encrypted image with classic AES	Chi-square of encrypted image with proposed method
4.1.04 in RGB	137865,3021	243,8203125	284,1979
4.1.04 in Red channel	65177,24219	223,6328125	227,5703
4.1.04 in Green channel	64178,57031	211,96875	304,8828
4.1.04 in Blue channel	112019,9766	269,359375	257,0625
4.1.05 in RGB	318137,1536	248,8177083	228,9531
4.1.05 in Red channel	253200,875	213,8046875	244,6953
4.1.05 in Green channel	296854,6406	279,25	216,7266
4.1.05 in Blue channel	388150,9453	263,1171875	300,5703
4.1.06 in RGB	130626,0729	255,3411458	264,2552
4.1.06 in Red channel	79835,30469	266,7734375	268,6094
4.1.06 in Green channel	52401,125	268,640625	257,7813
4.1.06 in Blue channel	127520,125	237,71875	214,7109
4.1.08 in RGB	441673,7891	252,1328125	282,7813
4.1.08 in Red channel	528543,5938	238,1796875	246,9453
4.1.08 in Green channel	338751,7734	286,46875	254,9844
4.1.08 in Blue channel	112098,3672	257,1640625	240,6563
4.2.03 in RGB	208272,9447	307,1705729	228,7891
4.2.03 in Red channel	82839,72852	268,8671875	229,1992
4.2.03 in Green channel	123352,0391	221,6953125	261,5137
4.2.03 in Blue channel	79942,61719	263,4609375	253,209
4.2.05 in RGB	2245319,335	225,858724	256,9245
4.2.05 in Red channel	652824,4922	271,8652344	271,7598
4.2.05 in Green channel	660991,3828	245,796875	240,8281
4.2.05 in Blue channel	1079186,006	255,4824219	279,4238
4.2.06 in RGB	204778,5195	256,3229167	247,8574
4.2.06 in Red channel	173145,3066	235,5566406	307,2891
4.2.06 in Green channel	124010,7168	262,703125	239,9141
4.2.06 in Blue channel	316923,5371	271,4648438	216,2793
4.2.07 in RGB	386627,2982	216,9733073	232,332
4.2.07 in Red channel	188611,2168	275,0136719	247,9473
4.2.07 in Green channel	299950,9297	180,2539063	258,9805
4.2.07 in Blue channel	464804,1777	253,4082031	261,4121

In the images, the Chi-square values are very high. The Chi-square values often sit in the hundreds of thousands. This study finds that the Chi-square values show that the pixel intensities are far from uniform. The pixel intensities in the images follow patterns and repeats. The patterns make the original images easy to break with attacks. In contrast, the Chi-square values for the encrypted images are low. The Chi-square values for the encrypted images using AES and the proposed method stay between two hundred and three hundred ten. It is possible to notice that the values are much closer to the expectation for a random distribution of 8-bit pixel values. The values suggest that the encryption schemes flatten the pixel intensity distribution. The proposed method works the same as AES. The proposed method sometimes shows uniformity. The Chi-square values are lower in "4.2.03 RGB" and "4.1.05 Green". The lower square values show that the pixel-level decorrelation works. The lower square values also show the strength of the proposed algorithm, in neutralizing statistical structures.

Fig. 7 shows the pixel intensity values, for the encrypted versions of a set of standard color images. The study tests two encryption techniques: Classic AES algorithm and the proposed encryption method. It is recorded that the measurements include the RGB images and the individual red, green and blue channels. It is recorded that the measurements cover the eight test images. Here are the images: "4.1.04 (Female)", "4.1.05 (House)", "4.1.06 (Tree)", "4.1.08 (Jelly beans)", "4.2.03 (Mandrill)", "4.2.05 (Airplane)", "4.2.06 (Sail-boat on Lake)" and "4.2.07 (Peppers)."

The average values differ in each image and, in each channel. The difference shows the variation, in the content the color distribution and the intensity range of unencrypted photos. The blue channel of "4.2.07" has low value. The red channel of "4.1.08" has high value. It is possible to notice that after encryption both methods normalize the pixel distributions. Both methods produce values that stay close, to 127.5 which's about the middle of the 0 to 255 grayscale range. This shows that the encrypted images have balanced pixel intensities. The encrypted images also have no bias toward any color or intensity range. It is possible to see that the proposed method works well as classic AES. The proposed method gives values that're a little closer to the ideal midpoint in several cases. For example, the proposed method is a little closer in "4.1.04 RGB" and, in "4.2.05 channel". This small improvement shows a level of consistency. The proposed method also shows that the input data is well balanced.

Table 3 shows a side by view of correlation coefficients for pixel intensities in encrypted color images. Two encryption algorithms are examined. The analysis includes RGB images and the red, green and blue channels, for each of eight test images. The test images are 4.1.04 (Female), 4.1.05 (House), 4.1.06 (Tree), 4.1.08 (Jelly beans), 4.2.03 (Mandrill), 4.2.05 (Airplane), 4.2.06 (Sail-boat on Lake) and 4.2.07 (Peppers).

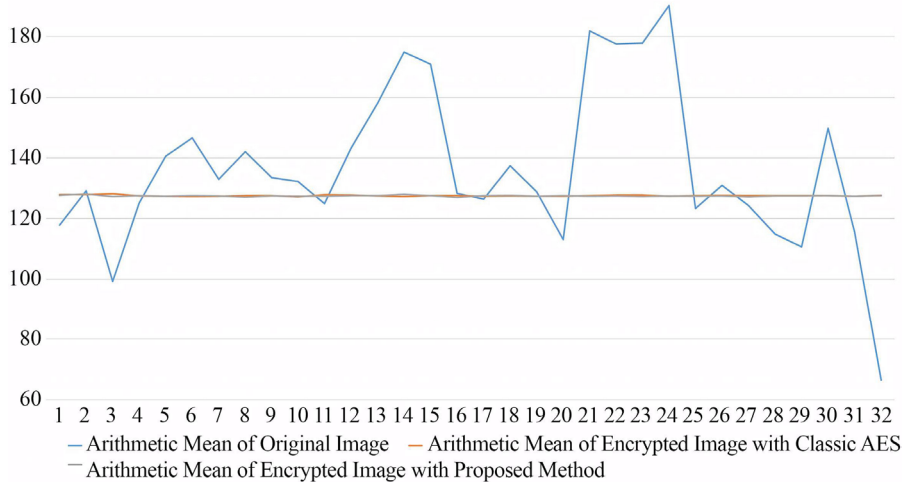


Fig. 7. Shannon entropy values of original and encrypted images using classic AES and the proposed encryption method

Table 3

Correlation coefficients of original and encrypted images using classic AES and the proposed method

Images	Correlation of original image	Correlation of encrypted image with classic AES	Correlation of encrypted image with proposed method
1	2	3	4
4.1.04 in RGB	0.606656904	0.000404481	–0.00212
4.1.04 in Red channel	0.975389519	0.002394226	0.006248
4.1.04 in Green channel	0.926010849	0.001397847	0.002602
4.1.04 in Blue channel	0.942402367	0.001486682	0.005043
4.1.05 in RGB	0.649988716	0.00345517	–0.00088
4.1.05 in Red channel	0.964214185	0.000626368	–0.00027
4.1.05 in Green channel	0.975968125	0.003694711	–0.00164
4.1.05 in Blue channel	0.977688984	–0.004532876	0.002901
4.1.06 in RGB	0.854412322	0.001109321	–0.00082

Continuation of Table 3

1	2	3	4
4.1.06 in Red channel	0.95850677	–0.008230011	–0.00314
4.1.06 in Green channel	0.96590147	0.006047482	0.003749
4.1.06 in Blue channel	0.960666281	0.00523664	0.001574
4.1.08 in RGB	0.496503683	–0.001942053	0.000502
4.1.08 in Red channel	0.97325021	0.003794497	–0.00164
4.1.08 in Green channel	0.970605978	0.004275545	–0.00329
4.1.08 in Blue channel	0.976703361	0.003356969	0.002725
4.2.03 in RGB	0.367528958	–0.000228143	–0.00102
4.2.03 in Red channel	0.921741162	–0.000846258	0.0034
4.2.03 in Green channel	0.864327583	0.000876849	–0.00153
4.2.03 in Blue channel	0.907116292	0.001132548	–0.00063
4.2.05 in RGB	0.827066556	0.000381378	–0.00025
4.2.05 in Red channel	0.972571996	0.000625845	0.000995
4.2.05 in Green channel	0.942478447	0.001839841	–0.00025
4.2.05 in Blue channel	0.963277802	0.001110037	0.000709
4.2.06 in RGB	0.808509123	0.000229872	–0.00043
4.2.06 in Red channel	0.954365206	8.43E–05	–0.0004
4.2.06 in Green channel	0.969151698	–0.000751655	–0.00211
4.2.06 in Blue channel	0.968950892	0.001163548	2.83E–05
4.2.07 in RGB	0.207585377	0.000233937	1.04E–05
4.2.07 in Red channel	0.961769758	–0.00094456	0.001321
4.2.07 in Green channel	0.977642763	–0.000569992	–0.00185
4.2.07 in Blue channel	0.962802871	–0.001096531	–0.00139

In the image, correlation coefficients are very high, in each color channel often above 0.95. This shows links between neighboring pixels. That is a feature of natural and structured image content. This high spatial correlation is a weakness, in image security because it can be used for attacks and image reconstruction. After encryption the correlation values drop dramatically. For both AES and the proposed method, correlation coefficients converge near zero. The encryption removes the built-in dependencies. In some cases, the values are a little negative. The encryption breaks the structure. The proposed method matches classic AES. It does a bit better. The proposed method moves correlation coefficients, toward zero or a little more negative on channels and images, for example "4.1.04 RGB" and "4.2.03 Green". The results show a level of decorrelation. The higher level of decorrelation gives the encryption resistance to analysis.

5. 3. Evaluation of randomness and diffusion/confusion properties

Table 4 shows the results that got by applying the NIST STS to evaluate the randomness of the expansion of Euler's number (*e*). The NIST STS includes tests that try to see if a binary sequence looks like a random sequence. The NIST STS tested the representation of e with 15 test categories. The NIST STS tested the representation of e with both single-parameter and multi-parameter subtests. Each test gave a p-value and a Boolean answer. The Boolean answer was True when the binary representation of e passed the randomness criteria.

Frequency test, block frequency test and runs test all gave p-values that're above the significance threshold (commonly $\alpha = 0.01$). The frequency test, block frequency test and runs test show no sign of non-randomness. the linear complexity test, the serial test and the approximate entropy test all passed. The linear complexity test, the serial test and the approximate entropy test support the idea that the bit stream of *e* does not have any fixed pattern. Discrete Fourier transform (spectral) Test gave p-values above 0.75. Cumulative sums (forward and backward) tests gave p-values above 0.75. That tells there is no component or cumulative bias. The random excursion test looks at how a cumulative random walk visits a state. The random excursion variant test looks at how a cumulative random walk visits a state. The random excursion test and the random excursion variant test both passed the states examined such, as −4 and +3. The Random excursion test and the random excursion variant test gave p-values above 0.75 for those states. That shows the walk behaves as expected. Every p-value stayed within the range. Every test conclusion returned True. The consistent success in the test set backs the claim that the binary expansion of e shows random behavior. The binary expansion of e fits the idea that irrational and transcendental numbers are normal. The normality of e is still not proved.

Table 5 shows the results of the NPCR test. The NPCR statistical test checks the sensitivity of the image encryption algorithm to changes, in the input images. The NPCR statistical test measures the percent of pixels that change in the encrypted image when a single pixel in the image is altered. The NPCR statistical test records the percent of changed pixels for three cases: the changed pixel is at the start of the image, the changed pixel is at the center of the image and the changed pixel is at the end of the original image.

Eight test images were evaluated ("4.1.04," "4.1.05," "4.1.06," "4.1.08," "4.2.03," "4.2.05," "4.2.06," and "4.2.07"). The NPCR values stayed near the ideal of 99.609%. The NPCR values fell between about 99.62% and 99.66%. The NPCR values show that the encryption process is very sensitive and spreads changes well.

Table 6 shows the UACI test results. The UACI test used to measure the intensity difference, between encrypted images. The encrypted images come from a one-pixel modification. The one-pixel modification was placed at three positions: the start, the center and the end of the plaintext images. The test to the Red channel, the Green channel and the Blue channel of each image was applied. Eight images were used: "4.1.04," "4.1.05," "4.1.06 " "4.1.08 " "4.2.03 " "4.2.05," "4.2.06 " and "4.2.07".

Table 4

NIST statistical test suite results for the binary expansion of Euler's number (e)

Test		P-value	
2.01. Frequency test:		(np.float64(0.026691502143413908), np.True_)	
2.02. Block frequency test:		(np.float64(0.5295752435509149), np.True_)	
2.03. Run Test:		(np.float64(0.7420869368737075), np.True_)	
2.04. Run test (Longest run of ones):		(np.float64(0.6587708207787935), np.True_)	
2.05. Binary matrix rank test:		(0.46894667787238087, True)	
2.06. Discrete Fourier transform (spectral) test:		(np.float64(0.7690237298394799), np.True_)	
2.07. Non–overlapping template matching test:		(np.float64(0.39519412554409317), np.True_)	
2.08. Overlapping template matching test:		(np.float64(0.11098741144722477), np.True_)	
2.09. Universal statistical test:		(np.float64(0.7548071751998757), np.True_)	
2.10. Linear complexity Test:		(np.float64(0.22054521140254396), np.True_)	
2.11. Serial test:		((np.float64(0.6356657503706866),np.True_), (np.float64(0.5183872851855769), np.True_))	
2.12. Approximate entropy test:		(np.float64(0.02890714536902641), np.True_)	
2.13. Cumulative sums (forward):		(np.float64(0.027479316739884543), np.True_)	
2.13. Cumulative sums (backward):		(np.float64(0.027479316739884543), np.True_)	
2.14. Random excursion test:			
STATE	xObs	P-value	Conclusion
'–4'	2.3578270958529184	np.float64(0.7977374869317202)	True
'–3'	2.4350857142857136	np.float64(0.7862379685231795)	True
'–2'	3.8589065255731922	np.float64(0.569903690430049)	True
'–1'	1.1428571428571428	np.float64(0.9502405577506894)	True
'+1'	2.714285714285714	np.float64(0.7439326487244222)	True
'+2'	4.705467372134038	np.float64(0.4528716348149052)	True
'+3'	1.291657142857143	np.float64(0.9357878072211877)	True
'+4'	1.896947700362944	np.float64(0.863212658888483)	True
2.15. Random excursion variant test:			
STATE	COUNTS	P-value	Conclusion
'–9.0'	19	np.float64(0.7705223744904057)	True
'–8.0'	11	np.float64(0.5575021253337239)	True
'–7.0'	22	np.float64(0.8240221313018803)	True
'–6.0'	36	np.float64(0.7472033251006591)	True
'–5.0'	37	np.float64(0.6884997410966343)	True
'–4.0'	34	np.float64(0.7618549862187101)	True
'–3.0'	31	np.float64(0.8577144806607083)	True
'–2.0'	29	np.float64(0.9385028847784713)	True
'–1.0'	25	np.float64(0.6884997410966343)	True
'+1.0'	29	np.float64(0.8936946693232327)	True
'+2.0'	22	np.float64(0.6434288435636205)	True
'+3.0'	22	np.float64(0.7199178531944465)	True
'+4.0'	40	np.float64(0.5444539772747027)	True
'+5.0'	51	np.float64(0.30559849700634445)	True
'+6.0'	52	np.float64(0.33355025411190464)	True
'+7.0'	49	np.float64(0.43638523105101057)	True
'+8.0'	33	np.float64(0.8630315773055417)	True
'+9.0'	22	np.float64(0.8458148386257579)	True

Table 5

NPCR test results for image encryption at different modification positions

Filename	In start	In center	In end
4.1.04	99.628703	99.623617	99.625651
4.1.05	99.634298	99.656677	99.636332
4.1.06	99.625142	99.642436	99.624634
4.1.08	99.632772	99.629720	99.640910
4.2.03	99.627940	99.626414	99.632136
4.2.05	99.635569	99.644725	99.647903
4.2.06	99.638875	99.643707	99.642944
4.2.07	99.638112	99.624888	99.637477

Table 6

UACI test results for image encryption at different modification positions

Filename	In start	In center	In end
4.1.04 RGB	22.216724	22.192171	22.242817
4.1.04 Red	33.326596	33.371833	33.427501
4.1.04 Green	33.417849	33.463296	33.495663
4.1.04 Blue	33.479357	33.507032	33.338719
4.1.05 RGB	22.288572	22.162013	22.230773
4.1.05 Red	33.373443	33.262287	33.417472
4.1.05 Green	33.635433	33.658603	33.430002
4.1.05 Blue	33.390317	33.385171	33.445955
4.1.06 RGB	22.205784	22.172981	22.173873
4.1.06 Red	33.351584	33.524379	33.448642
4.1.06 Green	33.494705	33.434759	33.363815
4.1.06 Blue	33.336654	33.670505	33.470405
4.1.08 RGB	22.369732	22.257078	22.333361
4.1.08 Red	33.515440	33.577085	33.536395
4.1.08 Green	33.611678	33.406252	33.531973
4.1.08 Blue	33.415462	33.532362	33.513602
4.2.03 RGB	22.222970	22.167413	22.232543
4.2.03 Red	33.444399	33.455022	33.468143
4.2.03 Green	33.439645	33.379735	33.477764
4.2.03 Blue	33.415108	33.455235	33.419195
4.2.05 RGB	22.217084	22.243738	22.242745
4.2.05 Red	33.481592	33.446135	33.505903
4.2.05 Green	33.437466	33.501389	33.537089
4.2.05 Blue	33.498408	33.575389	33.392965
4.2.06 RGB	22.259201	22.174928	22.265242
4.2.06 Red	33.446489	33.408080	33.517278
4.2.06 Green	33.490940	33.445001	33.518402
4.2.06 Blue	33.524925	33.545013	33.465042
4.2.07 RGB	22.270027	22.227893	22.270996
4.2.07 Red	33.558210	33.456928	33.606014
4.2.07 Green	33.498614	33.430952	33.463765
4.2.07 Blue	33.508133	33.462073	33.534172

In the RGB combined tests, UACI values are 22%. In the color channel tests, UACI values are close to the 33.33% ranging from 33.26 to 33.67%. The encryption algorithm creates brightness differences in the encrypted images even when the original image changes only a little.

5. 4. Occlusion attack stability assessment through block-wise data loss simulation

Table 7 shows the PSNR values, from encrypted image restoration experiments. Encrypted image restoration experiments tested two conditions. The first condition is occlusion called cropping. The second condition is salt-and-pepper noise injection. The evaluation used eight test images from "4.1.04", through "4.2.07". The evaluation used four cropping levels: 0.5, 0.25, 0.125 and 0.0625 fractions. The evaluation also used four salt-and-pepper noise levels: 0.01, 0.05, 0.1 and 0.2.

The PSNR values went up when used crop sizes. It shows the image quality got better because less information was hidden. When cropped the image by 50%, the PSNR values stayed 30.7 dB. That level is usually the point where the picture looks acceptable. When cropped the image down, to 6.25% the PSNR values rose a lot. The PSNR values reached 39.7 dB. That number tells the restored pictures look clear after part of the image was hidden. The encryption-restoration pipeline with a noise level of 0.01 was tested. The encryption-restoration pipeline kept the PSNR around 39.5 dB. When increased the noise to a level of 0.05 the PSNR fell to 33.1 dB. The PSNR at that level still showed noticeable but acceptable distortion. When pushed the noise to a level of 0.2 the PSNR dropped to 29.2 dB. The PSNR at that level still gave usable reconstruction quality even in very noisy settings. In these experiments, the algorithm kept PSNR values above the visibility threshold of 30 dB under moderate conditions. The algorithm showed that the encrypted image structure stayed stable. When the distortion was aggressive such as 0.5 cropping or 0.2 salt-and-pepper noise the PSNR values stayed in a range that allowed image recovery. The results prove that the encryption and decryption scheme is robust, against degradation, occlusion and impulse noise.

Table 7

PSNR values under occlusion and salt-and-pepper noise attacks for encrypted image restoration

Occlusion attack type	Image	PSNR result
1	2	3
0.5 cropped	4.1.04.	30.693529
0.5 cropped	4.1.05.	30.722073
0.5 cropped	4.1.06.	30.696840
0.5 cropped	4.1.08.	30.528555
0.5 cropped	4.2.03.	30.791410
0.5 cropped	4.2.05.	30.710397
0.5 cropped	4.2.06.	30.762512
0.5 cropped	4.2.07.	30.752243
0.25 cropped	4.1.04.	33.706083
0.25 cropped	4.1.05.	33.744029
0.25 cropped	4.1.06.	33.720116
0.25 cropped	4.1.08.	33.547943
0.25 cropped	4.2.03.	33.797596
0.25 cropped	4.2.05.	33.718105
0.25 cropped	4.2.06.	33.772428
0.25 cropped	4.2.07.	33.751945
0.125 cropped	4.1.04.	36.508002
0.125 cropped	4.1.05.	36.605489

Continuation of Table 7

1	2	3
0.125 cropped	4.1.06.	36.579137
0.125 cropped	4.1.08.	36.378206
0.125 cropped	4.2.03.	36.728631
0.125 cropped	4.2.05.	36.647911
0.125 cropped	4.2.06.	36.699537
0.125 cropped	4.2.07.	36.657404
0.0625 cropped	4.1.04.	39.503454
0.0625 cropped	4.1.05.	39.623624
0.0625 cropped	4.1.06.	39.591663
0.0625 cropped	4.1.08.	39.403559
0.0625 cropped	4.2.03.	39.739498
0.0625 cropped	4.2.05.	39.655637
0.0625 cropped	4.2.06.	39.696111
0.0625 cropped	4.2.07.	39.658147
0.01 Salt and Pepper	4.1.04.	39.468401
0.01 Salt and Pepper	4.1.05.	39.536186
0.01 Salt and Pepper	4.1.06.	39.501584
0.01 Salt and Pepper	4.1.08.	39.460564
0.01 Salt and Pepper	4.2.03.	39.464714
0.01 Salt and Pepper	4.2.05.	39.508769
0.01 Salt and Pepper	4.2.06.	39.485071
0.01 Salt and Pepper	4.2.07.	39.516729
0.05 Salt and Pepper	4.1.04.	33.175239
0.05 Salt and Pepper	4.1.05.	33.120223
0.05 Salt and Pepper	4.1.06.	33.229176
0.05 Salt and Pepper	4.1.08.	33.134326
0.05 Salt and Pepper	4.2.03.	33.165363
0.05 Salt and Pepper	4.2.05.	33.142427
0.05 Salt and Pepper	4.2.06.	33.183857
0.05 Salt and Pepper	4.2.07.	33.140128
0.1 Salt and Pepper	4.1.04.	30.930007
0.1 Salt and Pepper	4.1.05.	30.906321
0.1 Salt and Pepper	4.1.06.	30.913786
0.1 Salt and Pepper	4.1.08.	30.940815
0.1 Salt and Pepper	4.2.03.	30.916971
0.1 Salt and Pepper	4.2.05.	30.924651
0.1 Salt and Pepper	4.2.06.	30.901235
0.1 Salt and Pepper	4.2.07.	30.916583
0.2 Salt and Pepper	4.1.04.	29.260547
0.2 Salt and Pepper	4.1.05.	29.234637
0.2 Salt and Pepper	4.1.06.	29.248162
0.2 Salt and Pepper	4.1.08.	29.231981
0.2 Salt and Pepper	4.2.03.	29.253125
0.2 Salt and Pepper	4.2.05.	29.257453
0.2 Salt and Pepper	4.2.06.	29.255026
0.2 Salt and Pepper	4.2.07.	29.252438

Fig. 8 shows the robustness of the proposed image encryption and decryption method, against occlusion attacks. The figure shows reconstructed images that come from encrypted inputs that have been cut at four levels: 50%, 25%, 12.5% and 6.25%. The proposed image encryption and decryption method still works at each level. Each row, in Fig. 8, matches one occlusion level. Each row also holds a set of test images. The standard test images include 4.1.04 (Female), 4.1.05 (House), 4.2.03 (Mandrill) and other images that share the occlusion ratio.

It is possible to notice that the leftmost column shows the occlusion mask, in black and the matching ciphertext distortion. From left to right the reconstructed results show color fidelity, for each image as the occlusion level goes down. At 0.5 cropping, the decrypted images have distortion and noise. The decrypted images still keep some meaning. When the occluded area drops to 0.25, 0.125. 0.0625, the images get more detail and better visual quality. The perceptual restoration property rises a lot when the occlusion region's small. The diffusion strength of the encryption scheme shows how the encryption scheme reconstructs image content under perturbations. Fig. 8 validates the PSNR outcomes. Fig. 8 shows the restoration of image details even when ciphertext data is lost. Fig. 8 also illustrates recovery when data corruption occurs. The resilience to occlusion highlights the method's usability. The resilience to occlusion is crucial when data loss happens.

The resilience to occlusion also helps during data transfer, under conditions.

Salt-and-pepper noise was added to decrypted images to test the strength of an encryption method. The noise helps determine the error tolerance of the encryption method. When encrypted images travel through communication channels such, as networks or satellite connections, salt-and-pepper noise can affect the data. The noise flips bits at the pixel level. A decrypted image was tested with salt–and–pepper noise so that the evaluation matches conditions. A robust encryption system should get worse gradually. The encryption system should not fail completely when there is noise. Introducing salt-and-pepper noise at density levels prior to decryption enables evaluation of the algorithm's robustness. The densities tested include 0.01, 0.05, 0.1 and 0.2. This evaluation demonstrates if the algorithm can recover the images structure. It also indicates if the decryption is vulnerable to alterations, in the input. The assessment aids in understanding the scheme's ability to withstand damage. It also demonstrates the feasibility of image recovery under noisy conditions. Fig. 9 shows the scheme's reliability evaluation under a salt-and-pepper attack with different noise densities. The density values are 0.01, 0.05, 0.1, and 0.2. Each row of the image corresponds to one level of noise added to the ciphertext.

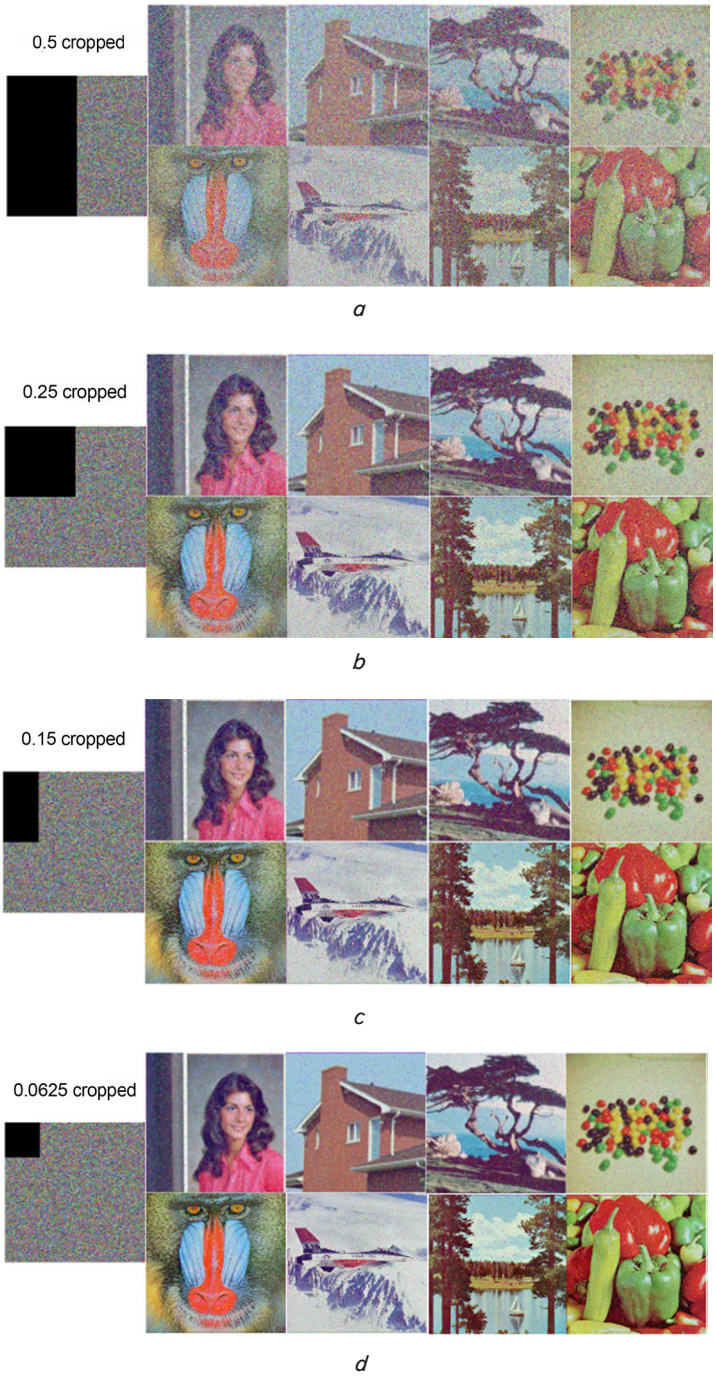
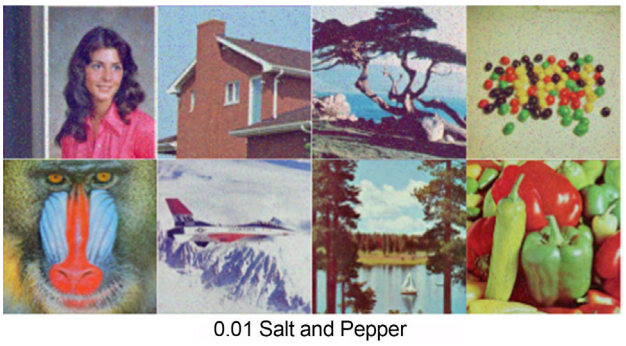


Fig. 8. Visual reconstruction of encrypted images under partial occlusion at different crop levels: *a* – encrypted image with 0.5 cropping; *b* – encrypted image with 0.25 cropping; *c* – encrypted image with 0.15 cropping; *d* – encrypted image with 0.0625 cropping



0.01 Salt and Pepper

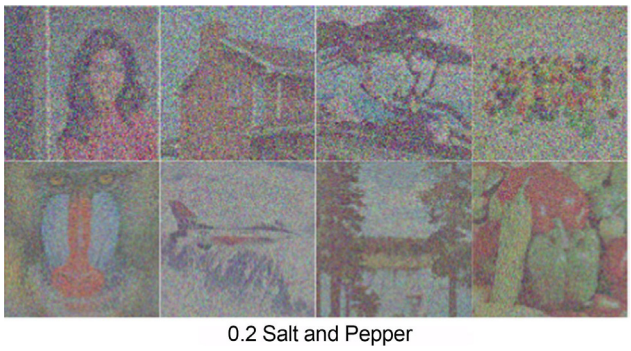
a

0.05 Salt and Pepper

b

0.1 Salt and Pepper

c



0.2 Salt and Pepper

d

Fig. 9. Visual impact of salt-and-pepper noise on decrypted images at varying noise densities: *a* — decrypted image with 0.01 salt and pepper; *b* — decrypted image with 0.05 salt and pepper; *c* — decrypted image with 0.1 salt and pepper; *d* — decrypted image with 0,2 salt and pepper

Examples of images: “4.1.04 Female”, “4.1.05 House”, “4.2.03 Mandrill”, “4.2.05 Airplane”, “4.2.06 Sailboat” and “4.2.07 Peppers”. The images sit left to right in each row. It is possible to notice that visual perception gets worse as the noise density goes up. At a noise density of 0.01, the decrypted images still keep the quality. The images have a noise. At a noise density of 0.05, graininess shows up. The images still look recognizable. In practice at a density of 0.1, the deterioration shows up. The deterioration is clear in areas. At a density of 0.2, the noise takes over. The image becomes hard to read. The dynamics show a drop in the result. The result drops as the noise rises. The algorithm reacts to data disturbances. The algorithm response shows the limits of the method in conditions. The noisy conditions appear often in communication channels. Examples: networks and damaged data carriers.

5. 5. Comparison of the proposed method with another approaches

Table 8 shows the compared correlation coefficients. The correlation coefficients were measured for the vertical direction and the diagonal direction. Encrypted versions of the Lenna image were used. The correlation coefficients close to zero show that adjacent pixels are not correlated. That result is good for image encryption systems because the method can break dependencies.

Table 8

Comparison of the correlation coefficient

Method	Horizontal	Vertical	Diagonal
Proposed method	0.001047	0.000513	0.001919
Finite State Machine based method [4]	−0.0066	−0.0106	−0.0014
Elliptic curve cryptography-based method using primitive polynomial [31]	−0.0022	−0.0123	−0.0067
Chaotic and hyperchaotic map-based method using enhanced S-box pixel permutator [32]	0.0484	0.0119	0.0141
Hyper-chaos-based method [33]	0.0059	0.0105	0.0142
Generalized Arnold map based method [34]	0.0173	−0.0027	−0.0177
Hyper-chaotic system with only one round diffusion process-based method [22]	0.0117	−0.0369	−0.0422
Spatiotemporal chaos process based method [35]	0.0143	0.0280	0.4466
Particle swarm optimization algorithm and cellular automata-based method [36]	0.0053	−0.0089	0.0126
Spatial bit-level permutation and high–dimension-based method [37]	−0.0574	−0.0035	0.0578
Constrained optimization algorithm [38]	0.0084	0.0089	−0.0011
Hybrid genetic algorithm [39]	0.0081	0.0093	0.0054
Tabu search algorithm [40]	0.0013	0.006	−0.0084
RNA-GA [41]	0.0011	0.0018	0.0018

The proposed method shows low correlation coefficients. The directional values are as follows: horizontal – 0.001047, vertical – 0.000513, diagonal – 0.001919. These values indicate pixel decorrelation. They also reflect the result of image encryption. These values are close to zero. They are compared with other encryption methods. For example, the finite-state machine-based method has coefficients of –0.0066, –0.0106, and –0.0014. The RNA-GA method has coefficients of 0.0011, 0.0018, and 0.0018. These methods also achieve

values near zero. However, the directional distribution is less uniform. Some methods show higher values. For example, the spatiotemporal chaos-based method has a diagonal correlation of 0.4466. This value indicates residual dependencies. It also reflects a decrease in encryption strength. Other methods also yield high coefficients. For example, a hyperchaotic system with one round of diffusion has values of −0.0369 vertically and −0.0422 diagonally. The bit-level permutation method has values of −0.0574 horizontally and 0.0578 diagonally. These results can reduce the security of the encrypted image. Table 9 shows a comparative evaluation of encryption methods. Two metrics are used for comparison: NPCR and UACI.

Table 9

Comparison of NPCR and UACI

Method	NPCR	UACI
Proposed method	99.654677	33.637433
DES [42]	99.5911	15.0325
RC4 [42]	99.5895	14.8831
RSA [42]	98.8525	45.0594
SIT [42]	99.5972	14.8598
Practical swarm optimization [43]	99.228	30.147
Hybrid genetic algorithm [39]	99.4981	32.8294
Genetic algorithm [44]	99.63	33.41
Tabu search algorithm [40]	99.6129	33.4127
RNA–GA [41]	99.6006	33.3479

The proposed method shows an NPCR value of 99.6547% and a UACI value of 33.6374%. These values indicate wide diffusion. They also demonstrate the method's ability to randomize the ciphertext image even with small changes in the plaintext. This result demonstrates the method's resistance to differential attacks. A comparison is made with other algorithms. For example, DES has a NPCR of 99.5911% and a UACI of 15.0325%. RC4 has an NPCR of 99.5895% and a UACI of 14.8831%. The proposed method shows a higher UACI value compared to these algorithms. This indicates a stronger response to changes in pixel intensity. RSA shows a UACI value of 45.0594%. This value is favorable for diffusion. However, RSA has a NPCR of 98.8525%. This combination may indicate incomplete sensitivity at the pixel level. Some algorithms belong to heuristic and bioinspired approaches. The genetic algorithm shows a NPCR of 99.630% and a UACI of 33.410%. The Tabu search method shows a NPCR of 99.6129% and a UACI of 33.4127%. These methods demonstrate good results but are inferior to the proposed method. The RNA–GA method shows an NPCR of 99.6006% and a UACI of 33.3479%. This result is also close to the values of the proposed approach.

6. Discussion of statistical indistinguishability and robustness of the surjective-FA-enhanced advanced encryption standard image encryption

Statistically random ciphertext results from two solutions. The first is replacing ShiftRows and MixColumns in AES with a surjective finite automaton. This automaton permutes bits at the state level. The second is using an S-box with high nonlinearity and limited differential homogeneity.

Fig. 1 shows the designed 16 × 16 S-box. It is generated using an algebraic and gradient approach. The generation criteria are: nonlinearity ≥ 100, algebraic degree ≥ 7, differential homogeneity ≤ 8, and no fixed or oppositely fixed points. This S-box increases confusion in encryption rounds. It affects the histograms and increases the entropy to approximately 8 bits per channel after encryption. This can be seen in Fig. 6. The finite automaton is $A = \langle Q, X, Y, \delta, \lambda \rangle$. It is constrained by the surjectivity requirement. It acts as a diffusion layer. It breaks the local spatial structure. The implementation of the block cipher is shown in the flowchart in Fig. 2. Encryption is performed in CBC mode. The process consists of 10 rounds. The key and block size are 128 bits. This combination of elements explains several observations. The correlation between adjacent pixels becomes close to zero in all directions. This is evident in Fig. 3, 4, and Table 3. Initially, the correlation exceeded 0.9. After encryption, it became approximately zero. This indicates the destruction of spatial redundancy. The RGB histograms become uniform. This is evident in Fig. 5. The Shannon entropy reaches values ≥ 7.99 bits across channels. This is shown in Fig. 6. The chi-squared values for the encrypted images are in the range of 210–310. For example, the value for "4.1.05 Blue" is 300.57. This range is consistent with a uniform 8-bit distribution. This is reflected in Table 2. Robustness tests show high PSNR values under conditions of partial data overlap and transmission noise. The PSNR is approximately 30.7 dB at 50% cropping. The PSNR is approximately 39.7 dB at 6.25% cropping. With salt-and-pepper noise, the values are approximately 39.5/33.1/29.2 dB at densities of 0.01/0.05/0.2. This means that the decryption process remains feasible even with degraded channel quality. The summary PSNR values are shown in Fig. 8.

The method involves replacing the AES linear diffusion with a state bit permutation based on FA. Previously, FA-based cryptosystems operated at the byte, word, or text file level. They provided ENT metrics, chi–square values, average values, and SCC. In this paper, the diffusion logic is combined with the developed S-box. Correctness is verified on color images. A comparison is made with AES on the same images. The proposed scheme shows an entropy of approximately 7.99–8.00. It also shows correlations close to zero. Some chi-square values approach the uniform range. This is evident in Tables 2, 3 and Fig. 6. A comparison with the literature shows that many chaotic or metaheuristic image ciphers preserve residual correlations. For example, the diagonal correlation can be around 0.45 in a spatiotemporal scheme. In the proposed method, correlations are close to zero in all directions and across all channels. NPCR values are approximately 99.655%. UACI values are approximately 33.64%. These results are comparable to strong baseline results. This is shown in Table 9. Previously, file-based FA methods, rather than image-based methods, demonstrated better entropy, chi-square, and SCC values compared to AES and 3DES. The test data size was approximately 5 KB. However, these studies did not utilize the pixel-to-pixel correlation metrics, NPCR, and UACI, applied in this work. This demonstrates the broad scope of the current study. In contrast to DES, RC4 and SIT [42], where UACI values are 14–15% this UACI is 33.637433%. UACI creates a change, in pixel intensities. The higher UACI means that even a tiny change, in the input image causes a spread of brightness differences. Larger spread makes differential analysis less effective. In contrast to [43], swarm-based methods have UACI that only reaches 30.147%, and this result has UACI that's close to the optimum of about 33%. The result shows a spread of inten-

sities across the whole cryptographic space. The diffusion structure is stable. This is made possible by the lack of gaps, in the luminance redistribution makes the result possible. In contrast to [39, 41, 44], genetic algorithms, including variants and RNA-GA give NPCR values from 99.4981%, to 99.63%, this method measured NPCR at 99.654677%, the highest among all methods. That means that changing one pixel in the image changes the maximum number of pixels in the encrypted image. This is made possible by a permutation-diffusion pipeline and, from using expressive nonlinear transformations.

One of the limitations of this study is that the application was demonstrated on RGB images from the USC-SIPI dataset. Generalization to other image types has not yet been performed. Examples of such types include medical grayscale images, HDR, and hyperspectral data. Compressed formats such as JPEG have also not been tested. Reproducibility depends on the S-box synthesis. The synthesis uses random NP exchanges. It also depends on the construction of the FA random state. Although the criteria are specified, repeated generation may yield different instances. The statistics of such instances may differ slightly. The security analysis in this paper is based on statistical metrics. Robustness is assessed using PSNR. Formal cryptanalysis is not presented. Linear paths, differential paths, related-key attacks, boomerang attacks, and chosen-ciphertext attacks are not considered. Error propagation and handling of initialization vectors in CBC mode create operational requirements.

The use of FA and a custom S-box increases the implementation complexity. This may create overhead compared to hardware-accelerated AES. Side-channel protection requires constant-time execution of operations. Comparison and optimization are also necessary. The description states that the AddRoundKey operation does not include XOR. This contradicts the AES semantics. This point requires clarification. If necessary, the algorithm should use reversible key mixing with constant execution time. This will avoid structural vulnerabilities. Current security tests include salt-and-pepper noise and framing. Additional tests could expand the evaluation. Examples of such tests include JPEG compression, Gaussian noise, filtering, and scaling. Adding such tests will provide a more complete picture of the method's robustness. This could lead to the use of lightweight authentication. For example, AEAD solutions compatible with FA could be considered.

The future development of this study includes:

1) formalization of diffusion and mixing of FA is required. Bounds such as the number of branchings and the probabilities of differential and linear attacks must be specified. Next, finding automata with such constraints is required. This problem is classified as NP-hard combinatorial design problems;

2) authenticated modes, such as FA-GCM, are considered. Stream cipher variants are also considered;

3) adaptive or image-based selection of S-boxes and FA is required. A synthesis tool is used for this. The goal is to find a balance between diversity and reproducibility;

4) hardware and SIMD implementations are required to restore throughput. This creates methodological challenges in preserving surjectivity and reversibility under parallelization.

7. Conclusions

1. Natural images showed dependence on neighboring pixels. Channel correlations often exceeded 0.9. AES reduced these correlations to values near zero. The proposed scheme also reduced correlations to values near zero. Sometimes, the values became negative. For example, for the image "4.2.03 (Mandrill)," the original red channel had $r \approx 0.9217$. After encryption, the value became $r \approx 0.0034$. The green channel had 0.8643 and became −0.0015. The blue channel had 0.9071 and became −0.0006. The average RGB value was 0.3675 and became −0.0010. Such values near zero indicate a loss of correlation. This also indicates resistance to statistical attacks. If consider three-dimensional spatial-chromatic dependencies, the encrypted data violated these relationships. Correlations shifted again from levels above 0.9 to values near zero, confirming that the structure across color planes disappears after encryption. For eight USC-SIPI images, the channel coefficients in the original versions were in the high range, for example, 0.975–0.978 for the red and blue channels. After encryption, the values were within thousandths of zero, for example, 0.0024, −0.0045, and 0.0015. When looked at the data, it is possible to see the effect, for both AES and the proposed method. The proposed scheme often gave the values.

2. When looked at the RGB histograms saw peaks and troughs that showed the pattern of intensity use. Encryption flattened the histograms in each channel. The combined distribution also flattened, giving a spread of pixel intensities. The flattening removed the patterns that could be analyzed. The entropy of the images was below 8.0 showing redundancy. Encryption raised the entropy to 8.0. The entropy of 8.0 in the RGB view and in each channel was observed. An entropy of 8.0 means randomness. It matches what strong encryption should produce. Looked at the values, in the original images and saw that the chi-square values were big up to hundreds of thousands. The distributions were not even. The square values dropped to a range of 200 to 310. The range is close to a distribution. In some cases, the proposed method gave a square value than AES. It is possible to see that eight images and their channels were tested for 4.2.03 RGB and, for 4.1.05 Green. It is found that the arithmetic mean of the encrypted data was 127.5. The value is at the midpoint of the [0.255] range. The result shows a balanced intensity distribution. The proposed method often gave a value that was nearer to the midpoint than the AES method did. For example, for 4.1.04 and 4.2.05, the proposed method gave a match. For example, the value for 4.1.06 was 0.8544. After AES, the value dropped to 0.0011. After the proposed method, the value dropped to −0.00082. It is also noticed that the RGB channels all behaved in the way.

3. All fifteen NIST STS categories were passed at $\alpha = 0.01$. The p-values were greater than 0.01. The NIST STS categories included frequency, block frequency, runs, spectral test (DFT) cumulative sums, linear complexity, sequential test and random variation test. The results showed outputs that were the same as random. The NPCR across eight images and three-pixel change positions was in the range of 99.623% to 99.657%. The NPCR is close 99,6%. The result confirms the distribution of variation when a single pixel changes. The UACI for RGB was 22%. The UACI for RGB matches the expected value for RGB. The UACI values by channel were ranged from 33.26 to 33.67%. The UACI values by channel are close to the value of 33.33%.

4. Under a 50% crop, the decrypted PSNR is 30.7 decibels. When the occlusion goes down to 25%, 12.5% and 6.25%, the PSNR rises to 39.7 decibels. When add salt and pepper noise, the PSNR stays at 39.5 decibels at a noise density of 0.01. The PSNR stays above 30 decibels when the noise density is

between 0.05 and 0.1. At a noise density of 0.2, the PSNR is 29.2 decibels and the image still show a recognizable structure.

5. When compare the proposed method, with ciphers and meta-heuristics, the proposed method gets the NPCR of 99.6547% and a UACI of 33.6374%. The proposed method beats DES, which has NPCR 99.5911% and UACI 15.0325. Beats RC4 has NPCR 99.5895% and UACI 14.8831. The proposed method also slightly beats the heuristics Tabu, which has NPCR 99.6129% and UACI 33.4127 and RNA-GA, which has NPCR 99.6006% and UACI 33.3479.

Conflict of interest

The authors declare that they have no conflict of interest in relation to this study, whether financial, personal, authorship or otherwise, that could affect the study and its results presented in this paper.

Financing

This work was supported by the Science Committee of the Ministry of Education and Science of the Republic of Kazakhstan under Grant No. AP19677422.

Data availability

Manuscript has no associated data.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the current work.

Authors' contributions

Alibek Barlybayev: Methodology, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Funding acquisition**; Zhanat Saukhanova:** Conceptualization, Methodology, Investigation, Writing – original draft, Supervision; **Gulmira Shakhmetova:** Conceptualization, Methodology, Formal analysis, Writing – original draft, Project administration**; Altynbek Sharipbay:** Conceptualization, Investigation, Writing – review & editing, Supervision, Funding acquisition**; Sayat Raykul:** Software, Validation, Resources, Data Curation**; Altay Khassenov:** Software, Validation, Resources, Data Curation, Visualization.

References

1. Chahar, S. (2024). Exploring the future trends of cryptography. Next Generation Mechanisms for Data Encryption, 234–257. https://doi.org/10.1201/9781003508632-16
2. Kong, J. H., Ang, L.-M., Seng, K. P. (2015). A comprehensive survey of modern symmetric cryptographic solutions for resource constrained environments. Journal of Network and Computer Applications, 49, 15–50. https://doi.org/10.1016/j.jnca.2014.09.006
3. Sharipbay, A., Saukhanova, Z., Shakhmetova, G., Barlybayev, A. (2023). Development of Reliable and Effective Methods of Cryptographic Protection of Information Based on the Finite Automata Theory. The Eurasia Proceedings of Science Technology Engineering and Mathematics, 26, 19–25. https://doi.org/10.55549/epstem.1409285
4. Shakhmetova, G., Barlybayev, A., Saukhanova, Z., Sharipbay, A., Raykul, S., Khassenov, A. (2024). Enhancing Visual Data Security: A Novel FSM-Based Image Encryption and Decryption Methodology. Applied Sciences, 14 (11), 4341. https://doi.org/10.3390/app14114341
5. Salami, Y., Khajevand, V., Zeinali, E. (2023). Cryptographic algorithms: A review of the literature, weaknesses and open challenges. Journal of Computer & Robotics, 2 (16), 63–115. http://dx.doi.org/10.22094/JCR.2023.1983496.1298
6. Hospodár, M., Jirásková, G. (2024). Conversions Between Six Models of Finite Automata. International Journal of Foundations of Computer Science, 36 (03), 321–344. https://doi.org/10.1142/s0129054124430020
7. Lotfi, Z., Khalifi, H., Ouardi, F. (2023). Efficient Algebraic Method for Testing the Invertibility of Finite State Machines. Computation, 11 (7), 125. https://doi.org/10.3390/computation11070125
8. Abubaker, S., Wu, K. (2013). DAFA - A Lightweight DES Augmented Finite Automaton Cryptosystem. Security and Privacy in Communication Networks, 1–18. https://doi.org/10.1007/978-3-642-36883-7_1
9. Kodada, B. (2022). FSAaCIT: Finite State Automata based One-Key Cryptosystem and Chunk-based Indexing Technique for Secure Data De-duplication in Cloud Computing. https://doi.org/10.36227/techrxiv.20443653.v1
10. Salas Pena, P. I., Ernesto Gonzalez Torres, R. (2016). Authenticated Encryption based on finite automata cryptosystems. 2016 13th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE), 1–6. https://doi.org/10.1109/iceee.2016.7751254
11. Khatua, K., Chattopadhyay, S., Dhar, A. S. (2025). Performance evaluation for accelerated and efficient prediction of different regression models aggravated with BPSO for enhancing area efficiency through state encoding in sequential circuits. Swarm and Evolutionary Computation, 95, 101919. https://doi.org/10.1016/j.swevo.2025.101919
12. Srilakshmi, S. (2012). On finite state machines and recursive functions application to cryptosystems. Anantapuram. Available at: https://shodhganga.inflibnet.ac.in/handle/10603/11436
13. Meskanen, T. (2001). On finite automaton public key cryptosystems. Turku Centre for Computer Science. Available at: https://www.finna.fi/Record/utu.998871095405971
14. Tao, R., Chen, S., Chen, X. (1997). FAPKC3: A new finite automaton public key cryptosystem. Journal of Computer Science and Technology, 12 (4), 289–305. https://doi.org/10.1007/bf02943149
15. Tao, R., Chen, S. (1999). The generalization of public key cryptosystem FAPKC4. Chinese Science Bulletin, 44 (9), 784–790. https://doi.org/10.1007/bf02885019
16. Kodada, B. B., D'Mello, D. A. (2021). Symmetric Key Cryptosystem based on Sequential State Machine. IOP Conference Series: Materials Science and Engineering, 1187 (1), 012026. https://doi.org/10.1088/1757-899x/1187/1/012026

17. Seitkulov, Y., Ospanov, R., Tashatov, N., Eraliyeva, B., Sisenov, N. (2023). Software tool for analysis and synthesis of cryptographic S-boxes. KazATC Bulletin, 126 (3), 257–266. https://doi.org/10.52167/1609-1817-2023-126-3-257-266
18. Ospanov, R., Seitkulov, Y., Eraliyeva, B. (2022). Generalized algebraic method for constructing 8-bit Rijndael S-block. KazATC Bulletin, 120 (1), 156–163. https://doi.org/10.52167/1609-1817-2022-120-1-156-163
19. Zheng, D., Alkawaz, M. H., Johar, M. G. M. (2025). Privacy protection and data security in intelligent recommendation systems. Neural Computing and Applications, 37 (34), 28431–28448. https://doi.org/10.1007/s00521-025-11189-3
20. Xu, X., Song, X., Liu, S., Zhou, N., Wang, M. (2025). New 2D hyperchaotic Cubic-Tent map and improved 3D Hilbert diffusion for image encryption. Applied Intelligence, 55 (7). https://doi.org/10.1007/s10489-025-06414-4
21. Dougherty, S. T., Klobusicky, J., Şahinkaya, S., Ustun, D. (2023). An S-Box construction from exponentiation in finite fields and its application in RGB color image encryption. Multimedia Tools and Applications, 83 (14), 41213–41241. https://doi.org/10.1007/s11042-023-17046-6
22. Norouzi, B., Mirzakuchaki, S., Seyedzadeh, S. M., Mosavi, M. R. (2012). A simple, sensitive and secure image encryption algorithm based on hyper-chaotic system with only one round diffusion process. Multimedia Tools and Applications, 71 (3), 1469–1497. https://doi.org/10.1007/s11042-012-1292-9
23. Dokku, N. S., David Amar Raj, R., Bodapati, S. K., Pallakonda, A., Reddy, Y. R. M., Krishna Prakasha, K. (2025). Resilient cybersecurity in smart grid ICS communication using BLAKE3-driven dynamic key rotation and intrusion detection. Scientific Reports, 15 (1). https://doi.org/10.1038/s41598-025-17530-z
24. Khan, M. A., Sharif, M., Akram, T., Raza, M., Saba, T., Rehman, A. (2020). Hand-crafted and deep convolutional neural network features fusion and selection strategy: An application to intelligent human action recognition. Applied Soft Computing, 87, 105986. https://doi.org/10.1016/j.asoc.2019.105986
25. Zhang, H., Wu, Q. J., Nguyen, T. M. (2013). Incorporating Mean Template Into Finite Mixture Model for Image Segmentation. IEEE Transactions on Neural Networks and Learning Systems, 24 (2), 328–335. https://doi.org/10.1109/tnnls.2012.2228227
26. Pareschi, F., Rovatti, R., Setti, G. (2012). On Statistical Tests for Randomness Included in the NIST SP800-22 Test Suite and Based on the Binomial Distribution. IEEE Transactions on Information Forensics and Security, 7 (2), 491–505. https://doi.org/10.1109/tifs.2012.2185227
27. Ullah, A., Jamal, S. S., Shah, T. (2017). A novel scheme for image encryption using substitution box and chaotic system. Nonlinear Dynamics, 91 (1), 359–370. https://doi.org/10.1007/s11071-017-3874-6
28. Abdul, Y., Ramasamy, V., Kukaram, G., Boulaaras, S., Alharbi, A. (2025). A dynamic image encryption scheme through 2-D cellular automata and chaotic logistic map. Scientific Reports, 15 (1). https://doi.org/10.1038/s41598-025-21225-w
29. Sharmila, S., Bhuvaneswaran, R. S., Vaithiyanathan, D. (2025). Secure image encryption using Rubik's Cube-based scrambling with chaos-driven diffusion and circular shifts. Optik, 339, 172533. https://doi.org/10.1016/j.ijleo.2025.172533
30. Hadžic, V., Bloem, R. (2024). Efficient and Composable Masked AES S-Box Designs Using Optimized Inverters. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2025 (1), 656–683. https://doi.org/10.46586/tches.v2025.i1.656-683
31. Sharma, P. L., Gupta, S., Nayyar, A., Harish, M., Gupta, K., Sharma, A. K. (2024). ECC based novel color image encryption methodology using primitive polynomial. Multimedia Tools and Applications, 83 (31), 76301–76340. https://doi.org/10.1007/s11042-024-18245-5
32. Kaushik, P., Attkan, A. (2021). A Chaotic and Hyperchaotic Map based Image Encryption Protocol for High-End Colour density Images using enhanced S-box pixel permutator. 2021 2nd International Conference on Computational Methods in Science & Technology (ICCMST), 174–180. https://doi.org/10.1109/iccmst54943.2021.00045
33. Hermassi, H., Rhouma, R., Belghith, S. (2011). Improvement of an image encryption algorithm based on hyper-chaos. Telecommunication Systems. https://doi.org/10.1007/s11235-011-9459-7
34. Ye, G., Wong, K.-W. (2012). An efficient chaotic image encryption algorithm based on a generalized Arnold map. Nonlinear Dynamics, 69 (4), 2079–2087. https://doi.org/10.1007/s11071-012-0409-z
35. Song, C.-Y., Qiao, Y.-L., Zhang, X.-Z. (2013). An image encryption scheme based on new spatiotemporal chaos. Optik - International Journal for Light and Electron Optics, 124 (18), 3329–3334. https://doi.org/10.1016/j.ijleo.2012.11.002
36. Zeng, J., Wang, C. (2021). A Novel Hyperchaotic Image Encryption System Based on Particle Swarm Optimization Algorithm and Cellular Automata. Security and Communication Networks, 2021, 1–15. https://doi.org/10.1155/2021/6675565
37. Liu, H., Wang, X. (2011). Color image encryption using spatial bit-level permutation and high-dimension chaotic system. Optics Communications, 284 (16-17), 3895–3903. https://doi.org/10.1016/j.optcom.2011.04.001
38. Su, Y., Tang, C., Chen, X., Li, B., Xu, W., Lei, Z. (2017). Cascaded Fresnel holographic image encryption scheme based on a constrained optimization algorithm and Henon map. Optics and Lasers in Engineering, 88, 20–27. https://doi.org/10.1016/j.optlaseng.2016.07.012
39. Abdullah, A. H., Enayatifar, R., Lee, M. (2012). A hybrid genetic algorithm and chaotic function model for image encryption. AEU - International Journal of Electronics and Communications, 66 (10), 806–816. https://doi.org/10.1016/j.aeue.2012.01.015
40. Abbasi, A. A., Mazinani, M., Hosseini, R. (2020). Evolutionary-based image encryption using biomolecules operators and non-coupled map lattice. Optik, 219, 164949. https://doi.org/10.1016/j.ijleo.2020.164949
41. Mahmud, M., Atta-ur-Rahman, Lee, M., Choi, J.-Y. (2020). Evolutionary-based image encryption using RNA codons truth table. Optics & Laser Technology, 121, 105818. https://doi.org/10.1016/j.optlastec.2019.105818
42. Ghaz, A., Seddiki, A., Nouioua, N. (2022). Comparative Study of Encryption Algorithms Applied to the IOT. The Eurasia Proceedings of Science Technology Engineering and Mathematics, 21, 469–476. https://doi.org/10.55549/epstem.1226679
43. Ahmad, M., Alam, M. Z., Umayya, Z., Khan, S., Ahmad, F. (2018). An image encryption approach using particle swarm optimization and chaotic map. International Journal of Information Technology, 10 (3), 247–255. https://doi.org/10.1007/s41870-018-0099-y
44. Kaur, M., Kumar, V. (2018). Beta Chaotic Map Based Image Encryption Using Genetic Algorithm. International Journal of Bifurcation and Chaos, 28 (11), 1850132. https://doi.org/10.1142/s0218127418501328