Dijlah Journal of Engineering Science (DJES) Vol. 3, No. 3, August, 2026, pp. 21-31
ISSN: Printed: **3078-9656, Online: 3078-9664**, paper ID: 183

RVGAC: A Round-Variable Authentication Code Based on a Hybrid Rijndael–Feistel Block Cipher for Network Security

Louay Flaieh Hasan [1], Yousif Louay Flaieh [2], Qahtan Mohammedadnan Azeez [3]
[1] Cybersecurity Science Department, College of Science, AlFarabi University, Baghdad, Iraq
[2] Department of Computer Science, College of Science, Dijlah University, Baghdad, Iraq
[3]Department of computer engineering techniques, College of Engineering Technology, University of Dijlah, Baghdad, Iraq

Article Info

Article history:

Received May.,5, 2026
Revised June.,2, 2026
Accepted Aug.,1, 2026

Keywords:

Block cipher
Symmetric encryption
Feistel structure
Rijndael
RVGAC
Round variability

ABSTRACT

The increasing frequency and sophistication of cyberattacks have made data confidentiality and integrity critical challenges in modern networked systems. Encryption algorithms, particularly symmetric-key block ciphers, form the foundation of contemporary cybersecurity infrastructures. Although well-established block ciphers such as AES provide strong security guarantees, their fixed round structures and static diffusion paths may introduce structural regularities when deployed in large-scale or long-term network environments.This paper proposes a novel symmetric-key block cipher, referred to as the Round Variable Generation Authentication Code (RVGAC), which integrates Feistel-based round swapping with Rijndael-inspired nonlinear substitution and linear diffusion mechanisms. The proposed design introduces controlled round-variable behavior, increasing cryptographic uncertainty and reducing exploitable statistical correlations. RVGAC achieves confusion through nonlinear FS-Boxes and S-Boxes, and diffusion through AddRoundKey, P-Vector permutation, MixColumns, and Feistel swapping.The security and randomness of the generated ciphertext sequences are evaluated using the NIST Statistical Test Suite and Strict Avalanche Criterion (SAC) analysis. Experimental results demonstrate strong statistical randomness and effective diffusion across multiple data types. These findings indicate that RVGAC is a suitable and robust block cipher for modern network security applications.

Corresponding Author:

Louay Flaieh Hasan
AlFarabi University
Dora, Almasafi street, Baghdad, Iraq
Email: Louay.hasan@alfarabiuc.edu.iq

1. INTRODUCTION

The protection of digital information is really important these days. In the past only governments and the military had to deal with data.. Now with the internet and cloud computing a lot of people have access to sensitive information. This means there is a risk of cyber threats and privacy breaches.To keep our information safe we need strong cryptographic mechanisms. One way to do this is by using key block ciphers. These are widely used to secure data when it is being sent or stored. There are two types of block cipher designs: Feistel networks and substitution-permutation networks.Feistel networks, like DES work by dividing the data into two halves and then swapping them. Substitution-permutation networks, like AES use a combination of substitution and permutation to secure the data. These designs are based on principles of confusion and diffusion.Over the years people have suggested new versions of these designs. Most of them still use fixed rounds, which limits their ability to diffuse the data. Some designs, like Rijndael use layers of substitution and permutation. These layers are usually the same for every round, which makes them less secure.

Recently some people have tried to combine elements of both designs to create something. Most of these designs still use fixed rounds and do not adapt to new attacks. This is a problem because it makes our digital information

more vulnerable.This paper introduces a design called the Round Variable Generation Authentication Code or RVGAC for short. RVGAC combines Feistel-based round swapping with substitution and linear diffusion mechanisms. This makes it more resistant to linear attacks.
The main contributions of this work are:

- A new hybrid architecture that combines Feistel and substitution-permutation networks. This creates a dynamic and secure design.
- A variable authentication mechanism that increases cryptographic uncertainty. This makes it harder for attackers to figure out how the data is being secured.
- New components that merge substitution with permutation-based spreading. This strengthens the security of the design.
- A rigorous security evaluation that uses tests and analysis to verify the security of the design.

This new design, RVGAC is a step forward in keeping our digital information safe. It addresses the limitations of designs and provides a more secure way to protect our data. The Round Variable Generation Authentication Code is an improvement over existing designs and has the potential to make a big impact in the field of cryptography. The RVGAC design is a dynamic architecture that combines the best of both worlds and its enhanced cryptographic uncertainty and diffusion-enhanced components make it a more secure option, for protecting sensitive digital information.

2. THEORITICAL FOUNDATIONS

To understand how RVGAC was designed we need to look at the ideas of cryptography that it is based on.

2.1 Confusion And Diffusion

Confusion and diffusion are the two fundamental principles governing secure cipher design, originally introduced by Shannon to describe the essential properties of secrecy systems [10]. Confusion aims to obscure the relationship between the ciphertext and the encryption key by introducing nonlinearity, while diffusion ensures that a single plaintext bit influences many ciphertext bits, thereby spreading statistical structure across the entire ciphertext [10], [11]. The proposed RVGAC algorithm enforces confusion through nonlinear FS-Box and S-Box substitution operations, which disrupt linear relationships between input data and round keys. Diffusion is achieved through a combination of linear transformations, including AddRoundKey, MixColumns, P-Vector permutation, and Feistel-based data swapping, ensuring that local changes rapidly propagate across the full data block [2], [7]. The integration of these nonlinear and linear components enhances resistance against statistical and linear cryptanalysis by increasing diffusion depth and reducing exploitable correlations across rounds.

2.2 Linear Cryptanalysis

Linear cryptanalysis is a known-plaintext attack that exploits linear approximations between plaintext bits, ciphertext bits, and key bits [12], [8]. Let $(x_0, \ldots, x_7)$ denote the input bits to an S-Box and $(y_0, \ldots, y_7)$ denote the corresponding output bits. A linear approximation exists if the relationship defined in Eq. (1) holds with a probability significantly different from 0.5:

$$\boldsymbol{x_{i1} \otimes x_{i2} \ldots \otimes x_{im} \otimes y_{i1} \otimes y_{i2} \ldots \otimes y_{in} = 0} \quad (1)$$

For an ideal S-Box, all such linear (and affine) relationships occur with probability exactly 0.5, indicating the absence of exploitable linear bias [12]. Any deviation from this value constitutes a linear approximation bias, which can be exploited by attackers to recover key information, particularly in the final rounds of block cipher encryption [12]. The proposed RVGAC algorithm mitigates linear cryptanalysis by employing strong nonlinear substitution mechanisms through FS-Boxes and S-Boxes, combined with round-variable diffusion paths. This combination disrupts linear correlation propagation across rounds and significantly reduces the feasibility of constructing effective linear approximations.

2.3 Feistel And Rijndael Structures

The general structure of a Feistel cipher is illustrated in Figure. 1, where the plaintext block is divided into two equal halves that are iteratively processed using round subkeys [2], [5]. In each round, one half of the data is transformed by a round function and combined with the other half using an XOR operation, followed by a swap operation. This structure guarantees invertibility regardless of the internal round function design.

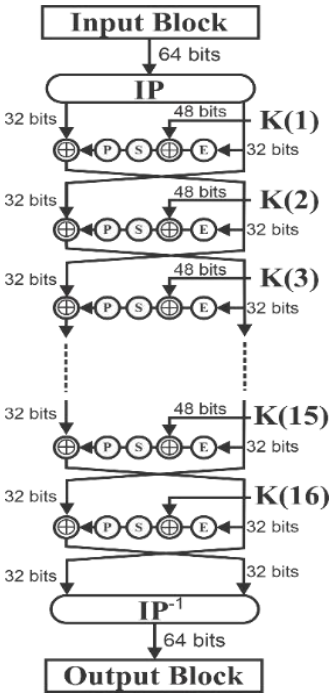

Figure 1. General structure of the Feistel encryption scheme

The Rijndael cipher, illustrated in Figure. 2, is based on a substitution–permutation network (SPN) operating on a 4×4 byte state matrix [2], [11]. Each round applies nonlinear substitution (SubBytes), linear diffusion (MixColumns), and key addition (AddRoundKey) operations to transform the state matrix. This design achieves strong confusion and diffusion properties and forms the basis of the Advanced Encryption Standard (AES).

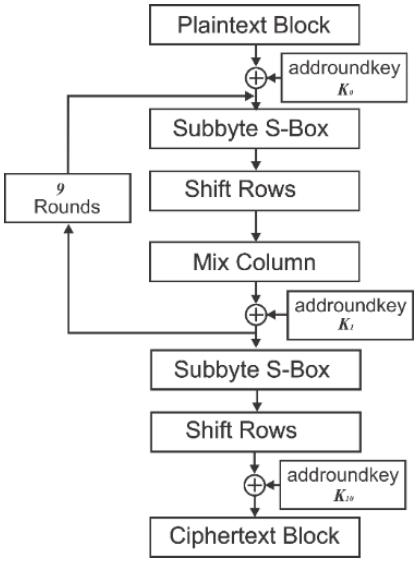

Figure 2. General encryption structure of the Rijndael block cipher

2.4 Counter Mode Of Operation

The counter (CTR) mode is a widely adopted block cipher mode of operation standardized and recommended by the National Institute of Standards and Technology (NIST) [2], [13]. In CTR mode, a sequence of counter values is encrypted using the block cipher and the secret key, and the resulting keystream blocks are XORed with the corresponding plaintext blocks to generate the ciphertext. Decryption follows the same process by XORing the ciphertext with the identical encrypted counter sequence [13].

The encryption and decryption processes of CTR mode are illustrated in Figure. 3(a) and Figure. 3(b), respectively. As shown in Figure. 3, the counter value is incremented for each block, ensuring keystream uniqueness and preventing block repetition.

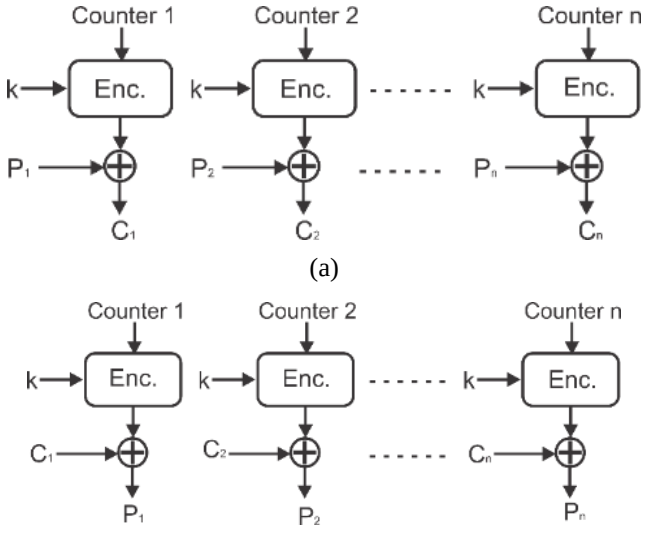

Figure 3. Counter (CTR) mode of block cipher operation: (a) encryption and (b) decryption

CTR mode offers several advantages, including high efficiency, parallelizable encryption and decryption, and resistance to pattern leakage, making it well suited for high-throughput and network-centric environments [2], [13]. For these reasons, the proposed RVGAC algorithm adopts CTR mode to ensure practical deployability while preserving strong security properties.

3. THE RVGAC ALGORITHM

3.1. General Structure

The overall encryption structure of the proposed RVGAC algorithm is illustrated in Figure. 4, each 256-bit plaintext block is divided into two equal halves of 128 bits. These halves are processed through multiple rounds following a Feistel-based structure, where one half is transformed by the round function while the other half is combined using an XOR operation. This iterative process continues for a predefined number of rounds, and the output of the final round produces the ciphertext block.

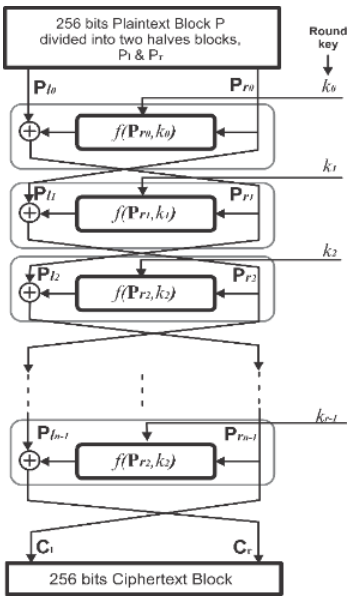

Figure 4. General encryption structure of the proposed RVGAC algorithm

3.2. Round Function Components

The internal components of the RVGAC round function are illustrated in Figure. 5. The round function is responsible for introducing confusion and diffusion at each encryption round and consists of a sequence of nonlinear and linear transformations inspired by the Rijndael design.

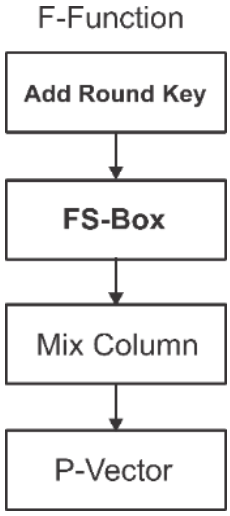

Figure 5. RVGAC round function components

3.3. FS-Box Substitution

The FS-Box substitution mechanism is illustrated in Figure. 6. In this operation, each input byte is divided into its most significant and least significant hexadecimal components, which are used to index the corresponding substitution values. This nonlinear substitution step plays a key role in enhancing confusion by disrupting linear relationships between plaintext bits and round keys.

MSHex FS-Box

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
6	5	7	4	8	9	D	2	F	B	3	E	1	A	0	C

LSHex FS-Box

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	3	8	5	C	E	F	D	B	A	2	6	0	9	4	7

Figure 6. FS-Box substitution mechanism based on hexadecimal MSB and LSB mapping

3.4. MixColumns Transformation

The MixColumns transformation, illustrated in Figure. 7, provides diffusion by mixing the bytes within each column of the state matrix. This operation is performed over the finite field $GF(2^8)$ using a fixed reversible matrix, ensuring that changes in a single byte propagate across multiple bytes in subsequent rounds.

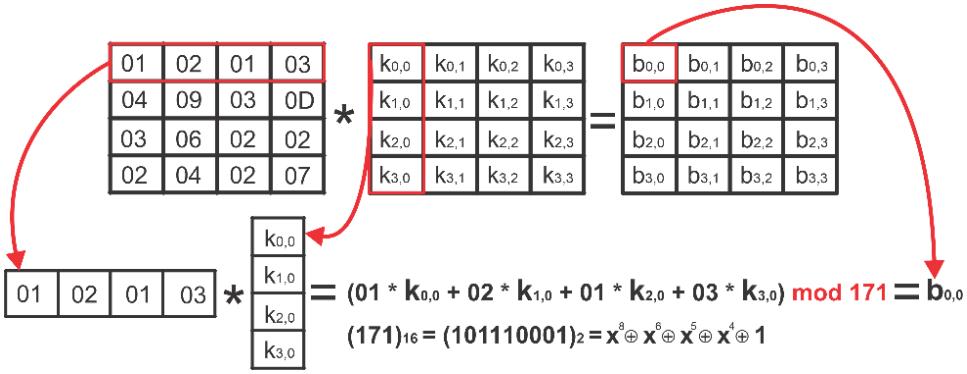

Figure 7. MixColumns transformation over GF(2^8)

3.5. P-Vector Permutation

To further enhance diffusion, RVGAC applies a P-Vector permutation to rearrange bit positions within the state matrix. The permutation used during encryption is defined in Table 1, while the inverse permutation applied during decryption is provided in Table 2.

Table 1. P-Vector permutation used in RVGAC encryption

i	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
P(i)	79	21	108	53	59	90	123	35	97	81	9	70	74	115	95	99
i	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
P(i)	118	18	51	77	29	103	58	54	94	89	111	72	23	69	92	3
i	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
P(i)	49	66	56	105	46	50	63	37	100	101	24	12	120	122	107	34
i	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
P(i)	85	45	106	75	20	55	127	5	10	104	28	7	84	82	91	39
i	**64**	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**	**75**	**76**	**77**	**78**	**79**
P(i)	64	33	87	16	11	102	67	116	68	80	83	13	114	88	41	73
i	**80**	**81**	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**	**90**	**91**	**92**	**93**	**94**	**95**
P(i)	1	62	22	78	65	30	19	0	52	44	121	124	27	6	57	31
i	**96**	**97**	**98**	**99**	**100**	**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**	**110**	**111**
P(i)	98	17	14	126	125	47	119	38	61	4	48	96	26	93	112	32
i	**112**	**113**	**114**	**115**	**116**	**117**	**118**	**119**	**120**	**121**	**122**	**123**	**124**	**125**	**126**	**127**
P(i)	76	86	15	40	25	113	36	8	110	2	71	60	43	42	117	109

Table 2. Inverse P-Vector permutation used during decryption

i	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
P(i)	87	80	121	31	105	55	93	59	119	9	56	68	43	75	98	114
i	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
P(i)	67	97	17	86	52	1	82	28	42	116	108	92	58	20	85	95
i	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
P(i)	111	65	47	7	118	39	103	63	115	41	125	124	89	49	36	101
i	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
P(i)	106	32	37	18	88	3	23	53	34	94	22	4	123	104	81	38
i	**64**	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**	**75**	**76**	**77**	**78**	**79**
P(i)	64	84	33	70	72	29	11	122	27	79	12	51	112	19	83	0
i	**80**	**81**	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**	**90**	**91**	**92**	**93**	**94**	**95**
P(i)	73	9	61	74	60	48	113	67	77	25	5	62	30	109	24	14
i	**96**	**97**	**98**	**99**	**100**	**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**	**110**	**111**
P(i)	107	8	96	15	40	41	69	21	57	35	50	46	2	127	120	26
i	**112**	**113**	**114**	**115**	**116**	**117**	**118**	**119**	**120**	**121**	**122**	**123**	**124**	**125**	**126**	**127**
P(i)	110	117	76	13	71	126	16	102	44	90	45	6	91	100	99	54

The complete processing flow of the RVGAC round function components, including FS-Box substitution, MixColumns transformation, and P-Vector permutation, is illustrated in Figure. 8.

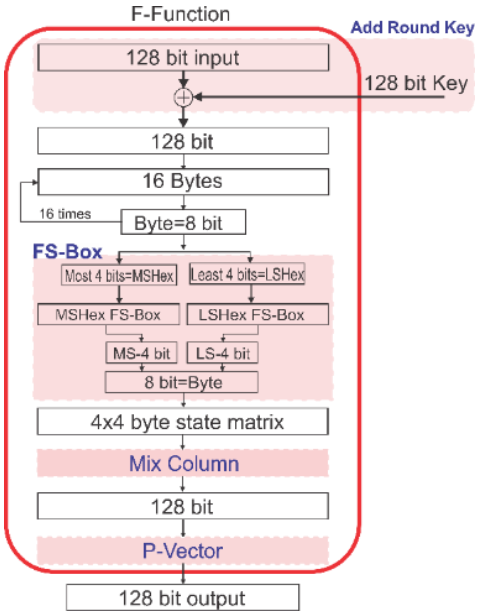

Figure 8. Processing flow of RVGAC round function components

3.6. Key Schedule

The key schedule in RVGAC is responsible for generating round keys from the primary secret key. The key expansion process is formally defined by Eq. (2) and Eq. (3).

The update operation for the first word $w[0]$ at round r is given by:

$$k[r]: w[0] = k[r-1]: w[0] \oplus SBox(k[r-1]: w[3] \gg 8) \oplus Rcon[r] \quad (2)$$

The remaining words are generated as follows:

$$k[r]: w[i] = k[r]: w[i-1] \oplus k[r-1]: w[i], \quad (3)$$

Where i=1, 2, 3

This key expansion process ensures round-dependent key diversity and strengthens resistance against related-key and linear cryptanalytic attacks. The overall key expansion and round-key generation process is illustrated in Figure. 9, while the generation of the round constants (Rcon) using polynomial multiplication over $GF(2^8)$ is shown in Figure. 10.

ISSN: Printed: **3078-9656, Online: 3078-9664**, paper ID: 183

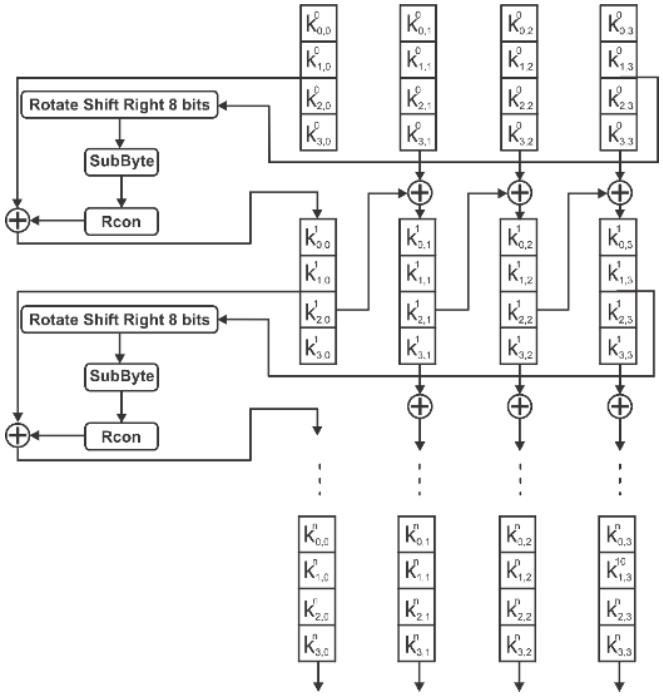

Figure 9. RVGAC key expansion and round-key generation process

$$x^0 = 1 \quad (01)_{hex}$$
$$x \times 1 = x \quad (02)_{hex}$$
$$x \times x = x^2 \quad (04)_{hex}$$
$$x \times x^2 = x^3 \quad (08)_{hex}$$
$$x \times x^3 = x^4 \quad (10)_{hex}$$
$$x \times x^4 = x^5 \quad (20)_{hex}$$
$$x \times x^5 = x^6 \quad (40)_{hex}$$
$$x \times x^6 = x^7 \quad (80)_{hex}$$
$$x \times x^7 = x^8$$
$$x^8 = 1 + x^4 + x^5 + x^6 \quad (71)_{hex}$$
$$x(1 + x^4 + x^5 + x^6) = x + x^5 + x^6 + x^7 \quad (E2)_{hex}$$
$$x(x + x^5 + x^6 + x^7) = x^2 + x^6 + x^7 + x^8$$
$$= x^2 + x^6 + x^7(1 + x^4 + x^5 + x^6)$$
$$= 1 + x^2 + x^4 + x^5 + x^7 \quad (B9)_{hex}$$
$$x(1 + x^2 + x^4 + x^5 + x^7) = x + x^3 + x^5 + x^6 + x^8$$
$$= x + x^3 + x^5 + x^6(1 + x^4 + x^6 + x^6)$$
$$= 1 + x + x^3 + x^4 \quad (1B)_{hex}$$

Figure 10. Rcon generation using polynomial multiplication

4. EXPERIMENTAL RESULTS AND ANALYSIS

4.1. NIST Randomness Tests Analysis

The NIST Statistical Test Suite (STS) was employed to evaluate the statistical randomness of the ciphertext sequences generated by the proposed RVGAC algorithm [14]. The NIST STS is a widely accepted evaluation framework for assessing the suitability of cryptographic algorithms by measuring their ability to produce statistically indistinguishable random sequences. The test results obtained for RVGAC across different input data types and sizes are summarized in Table 4. As shown in Table 3, all evaluated test cases satisfy the NIST acceptance criterion, where the computed p-values exceed the predefined significance level. These results indicate that the RVGAC algorithm produces ciphertext sequences with strong statistical randomness properties, consistent with the design objective of minimizing detectable structural patterns.

Table 3. NIST random ness test results for RVGAC sequences

N#	Test Name			PK	Text	Image	Audio	Video
1.	Monobit (Frequency)			SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
2.	Frequency Within Block			SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
3.	Runs			SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
4.	Longest 1's Run of Inblock			SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
5.	Rank of Binary Matrix			SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
6.	Spectral (DFT)			SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
7.	Matching of Nonoverlapping Template			SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
8.	Matching of Overlapping Template			SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
9.	Maurer (Universal Statistical)			SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
10.	Shortest LFSR (Linear Complexity)			SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
11.	Serial	Serial 1		SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
		Serial 2		SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
12.	Approxomate Entropy			SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
13.	Cumulative Sums	Forward		SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
		Backward		SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
14.	Random Excursion	At Point	(-4)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(-3)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(-2)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(-1)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(1)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(2)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(3)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(4)	FAIL	SUCCESS	SUCCESS	SUCCESS	SUCCESS
15.	Random Excursions (Variant)	At Point	(-9)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(-8)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(-7)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(-6)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(-5)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(-4)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(-3)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(-2)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(-1)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(1)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(2)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(3)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(4)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(5)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(6)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(7)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(8)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS
			(9)	SUCCESS	SUCCESS	SUCCESS	SUCCESS	SUCCESS

4.2. Avalanche Effect Analysis

In addition to statistical randomness testing, the avalanche effect was evaluated to assess the diffusion characteristics of the RVGAC algorithm. According to the Strict Avalanche Criterion (SAC), flipping a single input bit should result in approximately half of the output bits changing [10]. This property is essential for ensuring effective diffusion in block cipher designs.

The SAC and avalanche factor results obtained for RVGAC are presented in Table 6. As shown in the table, the algorithm exhibits a high degree of sensitivity to small changes in both plaintext and key values. These results confirm that RVGAC achieves strong diffusion behavior, ensuring that minor input variations lead to significant and unpredictable changes in the ciphertext.

Table 4. The throughput performance comparison between FBC and DES encryption

ndx	Ciphertext File Type	Size (Bits)	Avalanche Effect (Plaintext Change)	Avalanche Effect (Key Change)
1	Text	25400	0.5002	0.5000
2	Image	706360	0.4999	0.4999
3	Audio	2358584	0.5001	0.4998
4	Video	23108920	0.5001	0.5002

While these experimental results indicate strong statistical randomness and effective diffusion behavior, a formal cryptographic security proof for the proposed RVGAC algorithm is beyond the scope of this paper and is left for future work.

5. CONCLUSIONS

This paper presented RVGAC, a hybrid symmetric-key block cipher that integrates Feistel-based round swapping with Rijndael-inspired nonlinear substitution and linear diffusion mechanisms. By introducing controlled round-variable behavior, the proposed design increases cryptographic uncertainty and mitigates exploitable statistical correlations that may arise from fixed-round cipher structures. Experimental evaluation using the NIST Statistical Test Suite and Strict Avalanche Criterion (SAC) analysis demonstrates that RVGAC produces ciphertext sequences with strong statistical randomness and effective diffusion properties. These results confirm that the combination of nonlinear substitution, diversified diffusion paths, and round-dependent transformations contributes positively to the overall security characteristics of the algorithm. The security of RVGAC depends on the secrecy of the cryptographic key and the selected number of encryption rounds, with higher round counts providing increased resistance to cryptanalytic attacks at the cost of additional computational overhead. As future work, further analysis will be conducted to evaluate resistance against differential cryptanalysis and side-channel attacks, investigate hardware-accelerated implementations, and develop lightweight variants of RVGAC suitable for resource-constrained environments such as IoT systems. Future work will focus on providing a formal cryptographic security proof for RVGAC and extending the analysis to additional attack models, including differential and side-channel attacks.

REFERENCES

[1] Y. Zhong and J. Gu, “Lightweight block ciphers for resource-constrained environments: A comprehensive survey,” Futur. Gener. Comput. Syst., vol. 157, pp. 288–302, 2024.

[2] N. Mouha and M. Dworkin, Report on the block cipher modes of operation in the NIST SP 800-38 series. US Department of Commerce, National Institute of Standards and Technology, 2024.

[3] S. M. Al-Nofaie, S. Sharaf, and R. Molla, “Design trends and comparative analysis of lightweight block ciphers for IoTs,” Appl. Sci., vol. 15, no. 14, p. 7740, 2025.

[4] S. Aziz et al., “Next-Generation Block Ciphers: Achieving Superior Memory Efficiency and Cryptographic Robustness for IoT Devices,” Cryptography, vol. 8, no. 4, p. 47, 2024.

[5] A. Gour, S. S. Malhi, G. Singh, and G. Kaur, “Hybrid cryptographic approach: for secure data communication using block cipher techniques,” in E3S Web of Conferences, EDP Sciences, 2024, p. 1048.

[6] E. Almaraz Luengo and J. Román Villaizán, “Cryptographically secured pseudo-random number generators: Analysis and testing with NIST statistical test suite,” Mathematics, vol. 11, no. 23, p. 4812, 2023.

[7] M. S. Naik, M. Mallam, and C. S. Nataraju, “Machine learning-based lightweight block ciphers for resource-constrained internet of things networks: a review.,” Int. J. Electr. Comput. Eng., vol. 14, no. 3, 2024.

[8] Z. Liu, S. Han, Q. Wang, W. Li, Y. Liu, and D. Gu, “New insights on linear cryptanalysis,” Sci. China Inf. Sci., vol. 63, no. 1, p. 112104, 2020.

[9] L. Bassham et al., “A statistical test suite for random and pseudorandom number generators for cryptographic applications,” National Institute of Standards and Technology, 2008.

[10] C. E. Shannon, “Communication theory of secrecy systems,” Bell Syst. Tech. J., vol. 28, no. 4, pp. 656–715, 1949.

[11] J. Daemen and V. Rijmen, “Te Design of Rijndael: Te Advanced Encryption Standard (AES),” 2020, Springer Nature, NY, USA.

[12] M. Matsui, “Linear cryptanalysis method for DES cipher,” in Workshop on the Theory and Application of of Cryptographic Techniques, Springer, 1993, pp. 386–397.

[13] M. Bellare, J. Kilian, and P. Rogaway, “The security of the cipher block chaining message authentication code,” J. Comput. Syst. Sci., vol. 61, no. 3, pp. 362–399, 2000.

[14] S. H. Abdelhaleem, S. K. Abd-El-Hafiz, and A. G. Radwan, “Analysis and guidelines for different designs of pseudo random number generators,” IEEE Access, 2024.