ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal
Regular Issue, Vol. 14 (2025), e32318
eISSN: 2255-2863
DOI: https://doi.org/10.14201/adcaij.32318

Bit-Independent Criteria Evaluation of Custom S-Boxes: Enhanced AES Encryption Security

Amira S. El Batouty[a], Taymour A. Hamdalla[b], Heba A. Fayed[c], and Moustafa H. Aly[d]

[a] Electronics Department, Alexandria Higher Institute of Engineering and Technology, Alexandria, Egypt
[b] Physics Department, University of Tabuk, Tabuk, Saudi Arabia
[c] Electronics and Communications Engineering Department, Arab Academy for Science, Technology, and Maritime Transport, Alexandria, Egypt
[d] Electronics and Communications Engineering Department, Arab Academy for Science, Technology, and Maritime Transport, Alexandria, Egypt
✉ amira.elbatouty@aiet.eg.com, t-ahmed@ut.edu.sa, hebam@aast.edu, mosaly@aast.edu

KEYWORDS

security; encryption; S-Box; chaotic

ABSTRACT

Cryptography plays a pivotal role in safeguarding data from unauthorized access. Various encryption techniques have been developed and implemented to secure data during transmission through robust encoding measures across diverse systems, contributing to standardizing cryptographic practices. Previous studies have introduced several algorithms that focused on augmenting encryption security, such as the Key Scheduling Algorithm (KSA), S-Boxes customized to incorporate key, and PT dependencies within the RC4 algorithm. These S-Boxes were later expanded by integrating Henon Chaotic Maps and Logistic Chaotic Maps. The main objective was to create distinct S-Boxes with high-strengthening security features of the Advanced Encryption Standard (AES) algorithm. Different evaluations, such as tests for nonlinearity, Avalanche Effect (AE), the Strict Avalanche Effect (SAE), and Execution Time Performance Efficiency in percentage (ETPE %) were employed to assess the robustness of these S-Boxes.

This paper is dedicated to evaluating the robustness of different S-Boxes using Bit-Independent Criteria. We incorporated five distinct S-Boxes into the AES algorithm, presenting an innovative approach. The novelty of our work lies in applying Bit-Independent Criteria tests, quantitatively measuring the strength of each S-Box individually. Tests show that the S-Boxes produced through

chaotic maps noticeably strengthen encryption. The Henon based S-Box, using secret key "0123456789ABCDEF", achieved an average BIC value of -0.009, while the standard AES S-Box recorded -0.04, indicting a relative improvement of approximately 77.5 % in the output bit independence. Similar gains appeared with Logistic map-based S-Boxes under various key settings. By utilizing the Bit-Independent Criteria, our research aims to offer a comprehensive assessment of the S-Boxes' effectiveness in enhancing AES encryption security. This endeavor ultimately contributes to advancements in secure data transmission practices. Overall, our results demonstrate that chaotic S-Box design boosts AES security and guides future work on tougher cryptographic schemes.

1. Introduction

Cryptography plays a pivotal role in protecting data against unauthorized access, employing encryption techniques to secure data during transmission across various systems and to standardize cryptographic practices (Chari et al., 1999). Encryption ensures data integrity, enhances security, and protects data across different devices while preserving privacy. Commonly utilized encoding methods such as AES and DES play a crucial role in ensuring the security of information. To the best of our knowledge, no previous studies have specifically applied the Bit Independent Criterion (BIC) to evaluate customized S-Boxes integrated into the AES algorithm. This distinguishes our approach from prior research that focused primarily on nonlinearity and avalanche effect metrics.

Claude Shannon identified the fundamental concepts of diffusion and confusion as crucial for ensuring the security of any cipher system (Stallings, 2006). Diffusion conceals the relationship between ciphertext (CT) and plaintext (PT) by ensuring, each symbol in the CT is based on many symbols from the PT. Obscuration conceals the relationship between CT and the key, ensuring that a change in one bit of the key leads to changes in most or all bits of CT. The semi-field substitution boxes were investigated by Hussain et al. (2023) using permutation of symmetric group on a set of size 8 S_8. An effective procedure for generating S_8 semi-field substitution boxes with the same algebraic properties was established by Alharbi et al. (2023), where an innovative approach was proposed for constructing dynamic S-boxes using Gaussian distribution-based pseudo-random sequences. Cryptosystems are broadly categorized into two major classes: asymmetric key and symmetric key. Symmetric key encryption is additionally divided into two subdivisions, namely block ciphers and stream ciphers (Stallings, 2006). Among these, the S-Box stands out as one of the most critical components, serving as the sole nonlinear element that ensures the confusion characteristic in traditional block ciphers such as AES. The strength of this encryption algorithm depends on the meticulous design of a cryptographically robust S-Box (Shannon, 1998).

This paper focuses on the examination of the Bit Independent Criterion (BIC) for the evaluation of block cipher security, specifically by assessing the degree of confusion in elements resulting from the alteration of a single bit in the PT.

The following assumptions and limitations apply to our study.

- The evaluation is based on the standard AES block cipher structure, without incorporating hardware-specific optimizations or resistance to side-channel attacks.
- All S-Boxes are assumed to be invertible and have a fixed size of 16×16.

- Plaintexts (PTs) used during the BIC test are 128-bit blocks, where single-bit variations are introduced in a controlled manner.
- The encryption key size is fixed at 128 bits.
- The test environment is purely algorithmic and does not account for real-world conditions such as network latency, memory overhead, or concurrent processing scenarios.

The structure of this paper is presented as follows. Section 2 introduces a survey on different types of S-Boxes, while Section 3 explains the different kinds of security analysis. Section 4 delves into an in-depth study of the BIC test, while Section 5 analyzes the results of utilizing the BIC test with different types of S-Boxes. Finally, Section 6 concludes the paper.

2. Types of S-Boxes

S-Boxes serve as the sole nonlinear component within block cipher algorithms, playing a pivotal role in bolstering the confusion function of the encryption process. An S-Box takes an input of n bits and produces an output of m bits, with m and n not necessarily being equal. Crucially, an S-Box must be designed to be invertible for decryption purposes. A defining characteristic of S-Boxes is the absence of any discernible relationship between columns or rows.

2.1. The Static S-Box

1. Begin the initialization of the S-Box by arranging the byte values row by row in an ascending order.
2. Link every byte within the S-Box to its corresponding multiplicative inverse in the finite field GF (2^8) (Fahmy and Salama, 2005).
3. Finally, apply the Affine Transformation (AT) Equation to each bit of each byte in the S-Box as follows:

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (1)$$

Where each byte in the S-Box consists of 8 bits labelled (b_7,b_6,b_5,b_4,b_3,b_2, b_1, b_0), and the new S-Box consists of 8 bits labelled (b'_7, b'_6, b'_5, b'_4, b'_3, b'_2, b'_1,b'_0).

2.2. The Dynamic S-Box

The dynamic S-Box is an S-Box influenced by the encryption key with its functionality tied to the Key Scheduling Algorithm (KSA) Felicisimo et al., 2015). It can be explained as follows.

1. Initialization involves two arrays, both with 256 elements. The first array, S[256] is initialized with values ranging from 0 to 255, while the second array, K[256] is loaded with the shared secret key.
2. The commonly established secret key is divided into bytes and systematically mapped into the K array, one byte at a time.

$$T[i] = k\ [i \bmod \text{Key length}] \qquad (2)$$

3. S is a vector encompassing the scheduled key, while i and j denote the iterations within the round range. The result comprises 256 distinct values, entirely contingent on the input key.

$$j = (j+s[i] +t[i]) \bmod 256 \qquad (3)$$

4. AT is applied to the generated values (Felicisimo et al., 2015). This transformation serves the purpose of preventing fixed points and configuring the S-Box to be invertible.

2.3. Key and PT-Dependent S-Boxes in the RC4 Algorithm

Hosseinkhani and Haj et al. (2012) introduced S-Boxes dependent on both key and PT within the RC4 algorithm. In this context, the term "dependent S-Box" implies that the S-Box relies on the specific key and PT used in RC4 encryption, distinguishing it from the key and PT utilized in AES.

The stream cipher, RC4 encryption algorithm commonly employed in real-time communication scenarios (Stallings, 2006), relies on nonlinear data table modifications. It consists of two fundamental components: the Key Scheduling Algorithm (KSA) and the Pseudo-Random Generation Algorithm (PRGA). The dynamic S-Box section (Ejaz et al., 2021) delves into the KSA, while the PRGA is utilized to generate a single byte from the KSA. To visually illustrate the effectiveness of key and PT-dependent S-Boxes in the RC4 algorithm, please consult Figure 1.

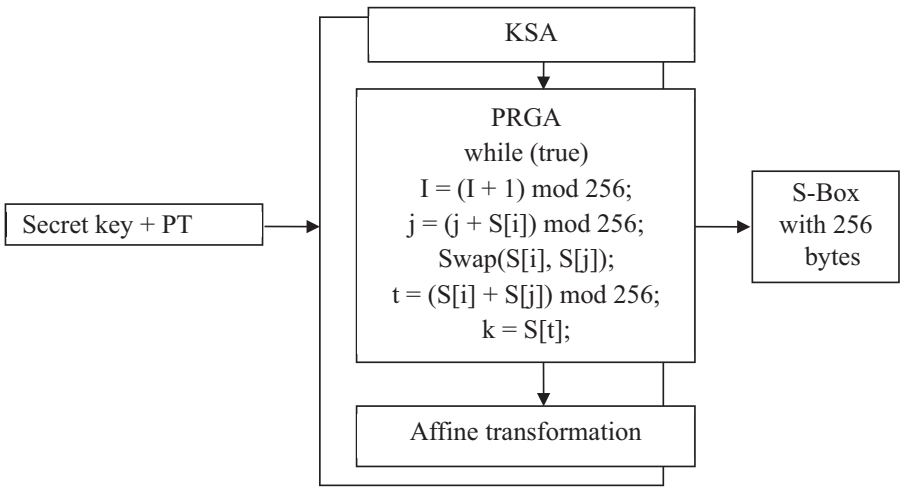

Figure 1. Block diagram depicting the key and PT-dependent S-Box within the RC4 algorithm, created by the authors on the basis of the study by El Batouty et al. (2019)

2.4. S-Boxes Customized for PT and Key in the RC4

The utilization of PT and key dependent S-Boxes within the RC4 algorithm involves a profound integration of key components, specifically PRGA and KSA. This integration results in the creation of robust S-Boxes, subsequently subjected to an AT to generate entirely new 256-byte S-Boxes. This strategic approach is adopted to enhance security against various forms of attacks (Agrawal and Himani, 2013; Ahmad and Hwang, 2015; El Batouty et al., 2020).

2.5. S-Boxes in the RC4 Algorithm Customized for PT and Key Using the Henon Chaotic Map

The RC4 algorithm utilizes the Henon chaotic map to enhance time efficiency and create innovative S-Boxes (Agrawal and Himani, 2013). By integrating two-dimensional Henon chaotic mapping into PRGA, the algorithm takes advantage of the complex dynamic behavior inherent in RC4 and the beneficial characteristics of chaotic systems. This integration of chaotic maps adds extra layers of data security by improving both confusion and diffusion, all the while reducing the algorithm's time consumption without compromising its security (El Batouty et al., 2020; Wen, 2014; El Batouty et al., 2019).

2.6. Key and PT-Dependent S-Boxes in the RC4 Algorithm Employing the Logistic Chaotic Map

The S-Boxes are derived through the integration of the Logistic Chaotic Map (LCM) into the RC4 algorithm to create key and PT-dependent S-Boxes. This process is executed within the PRGA phase, resulting in the development of a dynamic S-Box that is reliant on RC4 and enforced by the LCM (El Batouty et al., 2020).

In a prior study (Webster and Travares, 1998), the implementation of RC4 encryption involved the integration of the Henon chaotic map to improve time efficiency. The infusion of a chaotic signal characterized by a high degree of randomness into the RC4 algorithm amplifies both the confusion and diffusion effects within the encryption process (Agrawal and Himani, 2013). On the other hand, the LCM represents a simple yet distinctive algorithmic relationship that connects the output of one iteration to the input of the subsequent iteration, thereby further enhancing the cryptographic properties of the algorithm.

The mathematical equation for the LCM is provided as (Sosa, 2016):

$$x_{n+1} = r\, x_n\, (1 - x_n) \qquad (4)$$

Where r is a real management parameter within the range of 0 to 4, x_n denotes the initial value within the interval [0, 1], and n represents the number of iterations.

Integrating the one-dimensional LCM into the RC4 algorithm leverages the complex dynamic behavior of RC4 and the beneficial characteristics of chaotic systems. The process begins with the initiation of the KSA following the input of the key and PT. Following this, the execution of the PRGA is carried out in conjunction with the LCM. The resulting output comprises 256 bytes, each possessing unique values. Subsequently, an Affine Transformation is applied to derive the suggested S-Box, which consists of 256 bytes, as illustrated in Figure 2.

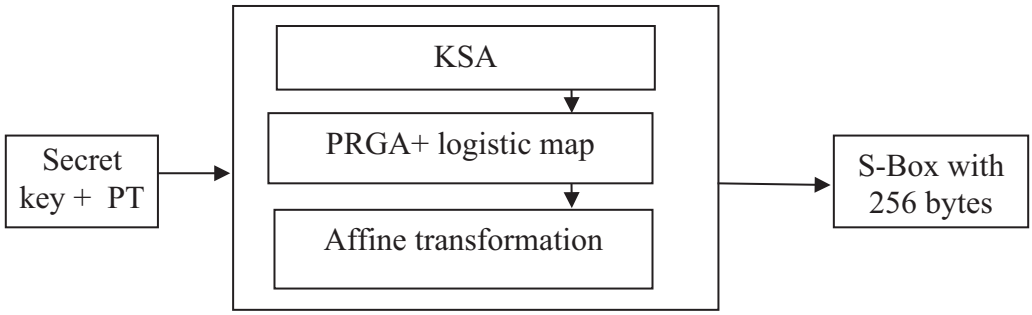

Figure 2. S-BOX victimization KSA and PRGA victimization LCM. Created by the authors on the basis of the study by El Batouty et al. (2020)

The distinctive feature setting apart this S-Box from the Dynamic S-Box lies in the use of the modified PRGA with the LCM for S-Box generation. The incorporation of the one-dimensional LCM into the RC4 algorithm aims to harness the complex dynamic behavior of RC4 and the advantageous properties inherent in chaotic systems. The process involves several distinct steps. Initially, the KSA is initialized on the basis of the provided input key and PT. Subsequently, the PRGA is executed with the integration of LCM. The resulting output comprises 256 distinct byte values. Following this stage, an Affine Transformation is applied to produce the proposed S-Box, which consists of 256 bytes, as illustrated in Figure 2. It is noteworthy that the S-Box generated in this approach differs from the Dynamic S-Box due to the utilization of a modified PRGA with LCM during the S-Box generation process. This innovative methodology enhances the cryptographic properties of the RC4 algorithm, combining the strengths of chaotic dynamics and traditional cryptographic techniques.

3. Security Analysis

This section discusses three key security analysis metrics that are essential in evaluating the strength of S-Boxes in cryptographic systems. These are: nonlinearity, avalanche effect, and execution time performance efficiency (ETPE). These criteria assess the confusion, diffusion, and efficiency properties, respectively. The PT sequence " 01, 11,…., FF" is chosen due to its structured format, which makes it suitable for tracking changes caused by single-bit modifications during testing. The overall time complexity for evaluating the BIC criterion for an S-Box is $o(n^2)$, where n is the number of PTs (typically 32 for each S-Box variant). The function o(.) gives the minimum number of calculation steps, leading to a minimum processing time. Each input triggers a full AES encryption, and the correlation coefficient computation involves linear scans over output vectors. Given five different keys, the total computational complexity of the full evaluation framework becomes $o(k \times s \times n^2)$, where k=3 keys, s=5 S-Boxes.

3.1. Nonlinearity

Nonlinearity is a key objective for S-box design, aiming to introduce a nonlinear transformation from the PT to the encrypted data. In this context, nonlinearity refers to achieving a probability of 0.5 for the number of bits inverted from input to output, indicating an equal likelihood for fifty percent of

all inputs and altered bits (Sosa, 2016; Hamdi, et al. 2015). While it is uncommon to precisely attain this probability, the goal is to approach or reach as close to 0.5 as feasible. The findings indicate that utilizing Henon chaotic maps in the S-box design enhances security, as evidenced by improved non-linearity test results.

The primary objective of the S-box lies in effecting a nonlinear transformation from the original PT to its encrypted counterpart. Nonlinearity, in this context, denotes a scenario where there is a 50 % probability of bits being inverted from the input to the output, implying that half of all input bits undergo a change (Sosa, 2016; Hamdi, et al. 2015). While achieving an exact 0.5 probability which is rare in practical applications, the goal is to approximate this probability as closely as possible. In reality, achieving precisely half probability is an infrequent outcome, prompting a focus on attaining a probability as near to 0.5 as practicable. Our investigation results indicate that employing Henon chaotic maps in the construction of the S-Box leads to heightened security, particularly evident in non-linearity tests. These findings suggest that the use of Henon chaotic maps in the S-Box yields superior nonlinearity characteristics when compared to alternative approaches.

3.2. Avalanche Effect

The avalanche effect characterizes an intriguing aspect of cryptanalytic algorithms (Eshmawi and Mahmoud, 2020). An S-Box is considered to exhibit the avalanche effect when a change occurs in one input bit of the PT, resulting in alterations in each output bit of the CT. This transformation occurs under the condition that at least half of the output bits will be flipped (El Batouty et al., 2020). Consider a new PT as follows: "01, 11, 22, 33, 44, 55, 66, 77, 88, 99, AA, BB, CC, DD, EE, FF".

The avalanche effect, a captivating characteristic of cryptanalytic algorithms (Eshmawi and Mahmoud, 2020), describes a phenomenon where an S-Box exhibits this effect when altering a single input bit in the PT results in every output bit in the CT undergoing a change, ensuring that at least half of the output bits are flipped (Ejaz et al., 2021). In the context of our study, we observe this effect with a new PT sequence:

"01,11,22,33,44,55,66,77,88,99,AA,BB,CC,DD,EE,FF".

$$\text{Avalanche effect} = (\text{NC}/\text{NT}) \times 100 \qquad (5)$$

Here, NC represents the count of changed bits in the CT and TN signifies the total number of bits in the CT. The Avalanche effect tests demonstrate that the S-Box utilizing LCM is more secure and exhibits a robust S-Box.

3.3. Execution Time Performance Efficiency (ETPE)

Speed serves as a gauge for assessing the relative performance enhancement during algorithm execution. The percentage Execution Time Performance Efficiency (ETPE) is defined as the following equation (Eshmawi and Mahmoud, 2020):

$$\text{ETPE } \% = \left(\frac{T_{old}}{T_{new}} - 1\right) \times 100 \qquad (6)$$

Where T_{old} is the average execution time of S-Box without modification and T_{new} is the average execution time of S-Box with modification.

Having reviewed the foundational security metrics of S-Boxes, we now proceed to a more advanced and focused evaluation method: the Bit Independent Criterion (BIC). This criterion offers a more granular analysis of how single-bit changes in the input affect the independence of output bits.

4. Bit Independent Criteria

In the context of BIC testing correlation coefficients are used to assess the degree of dependency between two output bits after a single-bit change in the input. Lower values (closer to 0) suggest higher independence between the bits, which reflects stronger diffusion properties and better cryptographic performance.

The Bit-Independent Criterion (BIC), initially introduced by "Webstar and Tavares" (Webster and Travares, 1998), represents a desirable cryptographic property. It serves to examine the nonlinear transformation behavior by modifying input bits, thereby inducing changes at the output bit level. This suggests that, for a vector set generated by the complement of a single PT bit, the entire avalanche variable must exhibit pair independence (José et al., 2020).

Webster and Tavares introduced a property known as the Bit Independence Criterion (BIC) for S-Boxes (Agrawal and Himani, 2013; Webster and Travares, 1998). A function f: $\{0, 1\}^n \rightarrow \{0, 1\}^n$; satisfies the BIC if for all i, j, k $\epsilon\{1,2,\ldots\ldots,n\}$, with j ≠ k, inverting input bit *i* leads to independent changes in output bits *j* and *k*. To quantify the bit independence concept, it is necessary to calculate the correlation coefficient between the j^{th} and k^{th} components of the output difference string, referred to as the avalanche vector, a^{ei}. A bit independence parameter corresponding to the effect of the i^{th} input bit change on the j^{th} and k^{th} bits of a^{ei} is defined as:

$$BIC\ (a_j; a_k) = \max_{1 \le i \le n} |\ corr(a_j^{ei}, a_{jk}^{ei})\ | \qquad (7)$$

Overall, the BIC parameter for the s-Box function f is obtained as:

$$BIC(f) = \max_{1 \le j,k \le n} BIC\ (aj; ak) \qquad (8)$$

$$j \neq k$$

To assess the adherence of f to BIC, one can examine the correlation coefficient, which is captured by BIC(f). This parameter varies between 0 and 1, with an optimal value of 0, while in the worst-case scenario, it reaches 1, which demonstrates how close f is to satisfying the BIC.

Figure 3 illustrates the flowchart detailing the BIC test. In an ideal scenario, the output of this test is expected to be precisely 0, signifying optimal performance. However, in the worst-case scenario, the output can reach a maximum value of 1.

5. Results and Discussion

The experimentation involved varying inputs with the alteration of only one bit, and the ensuing step involved assessing the correlation between the outputs when this singular bit was inverted. The analysis encompassed the application of the BIC test to evaluate the cryptographic properties. Specifically, for each of the five presented S-Boxes, 32 unique PTs were considered, where the sole point of

Procedure for BIC

```
correlation-Coefficient (float X[], float Y[], int n)
{
        sum_X = 0, sum_Y = 0, sum_XY = 0;
        squareSum_X = 0, squareSum_Y = 0;
        for ( i = 0; i < n; i++)
        {
                sum_X = sum_X + X[i];
                sum_Y = sum_Y + Y[i];
                sum_XY = sum_XY + X[i] * Y[i];
                squareSum_X = squareSum_X + X[i] * X[i];
                squareSum_Y = squareSum_Y + Y[i] * Y[i];
        }
        float corr = float(n * sum_XY - sum_X * sum_Y)/ sqrt((n * squareSum_X - sum_X *
sum_X)* (n * squareSum_Y - sum_Y * sum_Y));
        return corr;
}
```

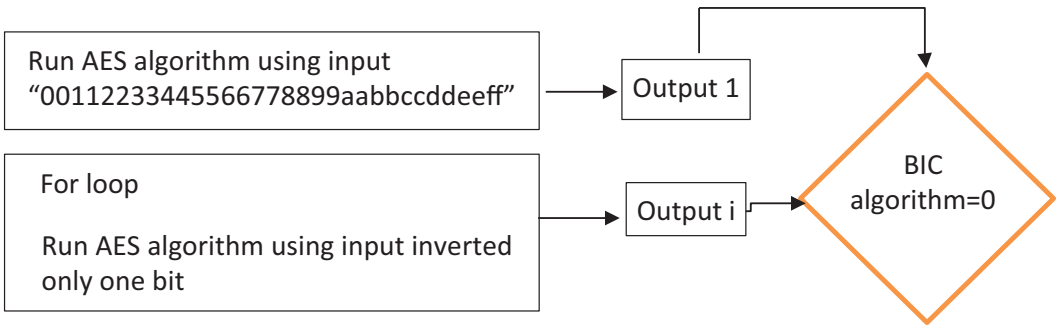

Figure 3. Flowchart for BIC test

distinction lay in the inversion of a single bit. The BIC test was systematically applied to all 32 possible PTs, and the outcomes of this evaluation are summarized in the subsequent tables, in Appendix, for different secret keys “0123456789ABCDEF”, “FEDCBA9876543210”, and “F9E8D7C6B5A4012”. Tables 1, 2, and 3 present measuring the average of BIC test for different secret keys and Figures 4, 5 and 6 represent the average of BIC test for different secret keys graphically. This study assumes a standard AES block cipher structure and does not incorporate hardware-specific factors or side-channel attacks.

Table 1. The average BIC

S-Box	Stand.	KSA	KSAPRGA	Henon	Logistic
BIC	-0.040079706	0.032841	0.049509	-0.009373	-0.04877

Table 2. The average BIC

Key	Standard	KSA	KSAPRGA	Henon	Logistic
Avg	0.021877	0.042928293	-0.03508	0.084707	-0.01473

Table 3. Average BIC

Key	Standard	KSA	KSAPRGA	Henon	Logistic
Avg	0.020304	0.003787	-0.00339	-0.01601	0.001217

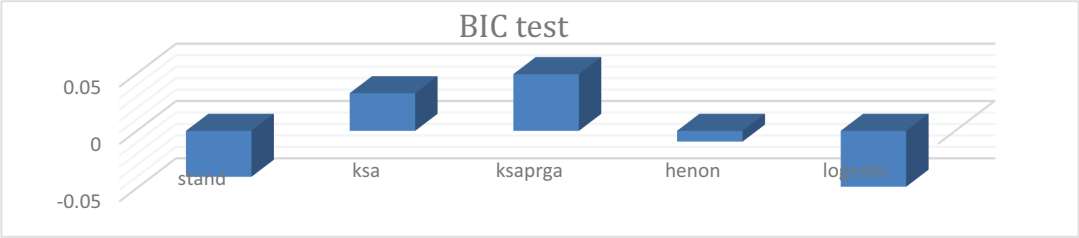

Figure 4. Average BIC test analysis for key 0123456789ABCDEF

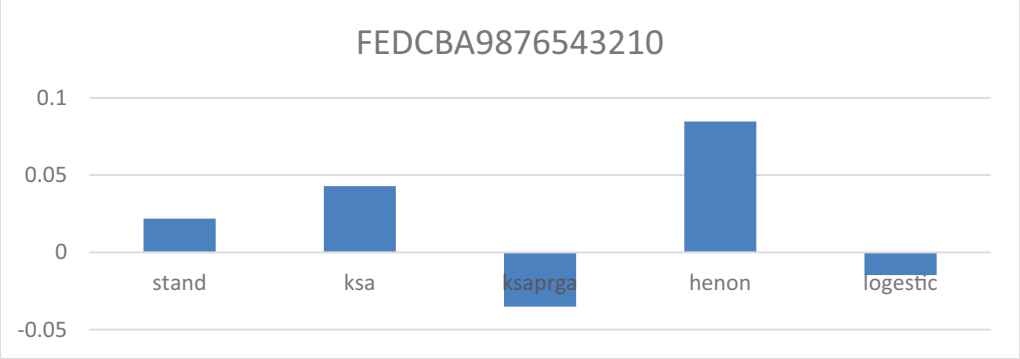

Figure 5. Average BIC test analysis for key FEDCBA9876543210

Tables 1-3 and Figures 4-6 are quite extensive, a general analysis is now offered and comments on several key aspects.

1. Correlation values:
   - o The correlation values in the tables represent the relationship between the outputs when changing only one bit in the input.
2. Variation across bit position:
   - o One noticeable trend is the variation in correlation values across different bit positions. Some bit positions lead to higher correlations, indicating that flipping those bits has a more predictable impact on the CT.

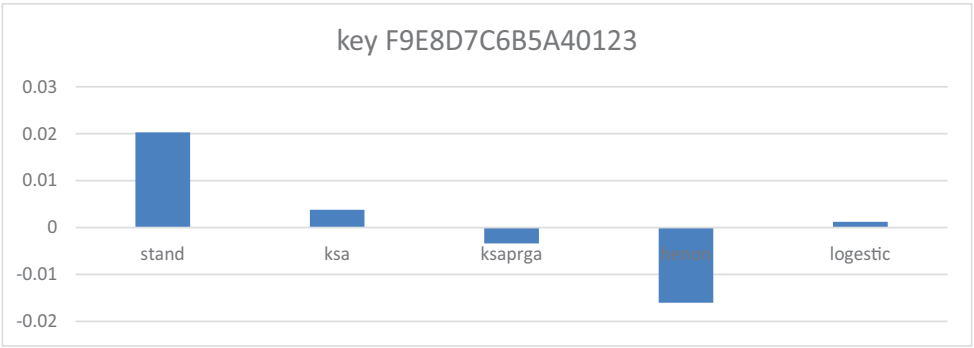

Figure 6. Average BIC test analysis for key F9E8D7C6B5A40123

Comparing the results, the key and PT dependent S-Box victimization RC4 algorithm implemented using Henon chaotic map gives the least correlation between the outputs. That means, the utilization of the Henon chaotic map in implementing the key and PT-dependent S-Box in the RC4 algorithm results in enhanced security compared to the key and PT-dependent S-Box in the standard RC4 algorithm.

6. Conclusion

The BIC serves as a valuable tool for scrutinizing the nonlinear transformation dynamics induced by alterations in input bits, influencing the corresponding output bits. In an ideal scenario, BIC registers a value of 0, indicating optimal performance, while its worst-case manifestation reaches a value of 1. A comparative analysis of the results reveals noteworthy findings for both the standard S-Box and the dynamic S-Box, where the BIC consistently hovers around zero or even below, denoting an absence of correlation between output variations and the inversion of a single input bit.

Contrastingly, the key and PT-dependent S-Box, implemented within the RC4 algorithm, presents a distinct behavior. When the input is set to "000122334455667788 99aabbccddeeff", the correlation manifests in its worst-case scenario. However, intriguingly, using any other input yields no discernible correlation between output variations. Further exploration into the application of the key and PT-dependent S-Box within the RC4 algorithm, implemented through the Henon chaotic map, reveals the least correlation between output changes. This observation implies that the implementation of the key and PT-dependent S-Box using the RC4 algorithm and Henon chaotic map is more secure compared to its counterpart utilizing the standard RC4 algorithm.

The correlation outcomes exhibit variations depending on the AES secret key employed. Specifically utilizing the key "0123456789ABCDEF", the S-box generated by Henon Chaotic map demonstrates a minimal BIC value of -0.009373, surpassing the performance of AES with the standard S-Box. Similarly, when the key is set to "FEDCBA9876543210", the S-Box generated by logistic chaotic maps yields a minimal BIC value of -0.01473. Furthermore, with the key "F9E8D7C6B5A40123", the S-Box generated by RC4 attains a minimal BIC value of -0.00339.

To summarize, the BIC test results for the AES encryption algorithm, generated by different chaotic maps or RC4, consistently outperform the standard S-Box when considering various secret keys. These findings emphasize the potential advantages of incorporating diverse chaotic maps or

the RC4 algorithm in the generation of S-Boxes for enhanced security within the AES encryption algorithm.

Future research could extend the BIC evaluation framework to real time encryption scenarios and IoT-constrained environments. This includes exploring adaptive or AI-driven dynamic S-Box generation techniques, applying the methodology to other cryptographic primitives such as lightweight ciphers or stream ciphers, and evaluating hardware implementations and side-channel resistance in future studies. These extensions aim to broaden the applicability and robustness of the proposed approach.

References

Agrawal, B., & Himani, A. (2013). Implementation of AES and RSA using chaos system. *International Journal of Scientific and Engineering Research, 4*(5) 1413-1417. https://www.semanticscholar.org/paper/Implementation-of-AESand-RSA-Using-Chaos-System-AgrawalAgrawal/0a200eca5f70d0f597ce0ac747ca40f30ce4b904

Ahmad, J., & Hwang, S. O. (2015). Chaos-based diffusion for highly autocorrelated data in encryption algorithms. *Signal Image Video Process, 13*(82) 1839–1850. https://doi.org/10.1007/s11071-015-2281-0

Alharbi, A. R., Jamal, S. S., Khan, M. F., Gondal, M. A., & Abbasi, A. A. (2023). Construction and optimization of dynamic s-boxes based on Gaussian distribution. *IEEE Access, 11*, 35818-35829. https://doi.org/10.1109/ACCESS.2023.3262313

Chari, S., Jutla, C., Rao, J. R., & Rohatgi, P. (1999). *A cautionary note regarding evaluation of AES candidates on smart-cards*. Proceeding of Second Advanced Encryption Standard (AES) Candidate Conference, Rome, Italy, pp. 1-15. https://doi.org/10.1.1.38.9312

Ejaz, A., Shoukat, I. A., Iqbal, U., Abdul, R., & Kanwal A. (2021). A secure key dependent dynamic substitution method for symmetric cryptosystems. *PeerJ Computer Science, 7*, Article e587. https://doi.org/10.7717/peerj-cs.587

El Batouty, A. S., Farag, H. H., Mokhtar, A. A., El-Badawy, E. A., & Aly, M. H. (2020). Improvement of radio frequency identification security using new hybrid advanced encryption standard substitution box by chaotic maps. *Electronics, 9*(7), Article 1168. https://doi.org/10.3390/electronics9071168

El Batouty, A. S., Farag, H. H., Mokhtar, A. A., & El-Badawy, E. A. (2019). *New hybrid AES static s-box algorithm using chaotic maps*. Proceedings of the 2019 IEEE International Conference on Information Technologies (InfoTech-2019), Constantine and Elena, Bulgaria, pp. 1-5. https://doi.org/:10.1109/InfoTech.2019.8860892

Eshmawi, A. A., & Mahmoud, E. E. (2020). Secure communications via complex phase synchronization of pair complex chaotic structures with a similar structure of linear terms with modifying in nonlinear terms. Alexandria *Engineering Journal, 59*. https://doi.org/10.1016/j.aej.2020.01.003

Fahmy, F., & Salama, G. (2005). *A proposal for key-dependent AES*. Proceeding of 3rd International Conference: Sciences of Electronic, Technologies of Information and Telecommunications (SETIT2005), Tunisia, pp. 1-7. http://www.setit.rnu.tn/last_edition/setit2005/image-video/53.pdf

Felicisimo, V., & Wenceslao, Jr. (2015). Performance efficiency of modified AES algorithm using multiple S-boxes. *International Journal of New Computer Architectures and their Applications, 5*(1) 1-9. https://doi.org/10.17781/P001620

Hamdi, M., Rhouma, R., & Belghith, S. (2015). A very efficient pseudo-random number generator based on chaotic maps and S-box tables. *International Journal of Electronics and Communication Engineering, 9*(2) 481-485. https://doi.org/10.5281/zenodo.1100418

Hosseinkhani, R., & Javadi, H. H. S. (2012). Using cipher key to generate dynamic S-Box in AES cipher system. *International Journal of Computer Science and Security, 6*(1) 19-28. https://www.cscjournals.org/manuscript/Journals/IJCSS/Volume6/Issue1/IJCSS630.pdf

Hussain, I., Anees, A., & Al-Maadeed, T. A. (2023). A novel encryption algorithm using multiple semifield S-boxes based on permutation of symmetric group. *Computational and Applied Mathematics, 42*, Article 80. https://doi.org/10.1007/s40314-023-02208-x

José, E., Capó, M., Miguel, C., Pérez, L., Rojas, O., Gómez, G. S., & Llanes, R. S. (2020). Bit independence criterion extended to stream ciphers. *Appl. Sci., 10*(21). Article 7668. https://doi.org/10.3390/app10217668

Shannon, C. (1998). Communication theory of secrecy systems. Bell Systems *Technical Journal, MD Computing, 15*, 57-64. https://doi.org/10.1002/j.1538-7305.1949.tb00928.x

Sosa, P. M. (2016). *Calculating nonlinearity of Boolean functions with Walsh Hadamard transform*. UCSB, Santa Barbara, CA-USA, pp. 1-4. http://konukoii.com/blog/wp-content/uploads/2016/06/FinalPaper.pdf

Stallings, W. (2006). *Cryptography and Network Security, Principles and Practices*. 4th Ed. Pearson Education, Prentice-Hall of India Pvt. Limited, India. https://dl.acm.org/doi/book/10.5555/280574

Webster, A. F., & Travares, S. E. (1998). *On the design of S-boxes*. Queen's University Kingston, Springer-Verlag, Canada. https://doi.org/10.1007/3-540-39799-X_41

Wen, H. (2014). *A review of the Henon map and its physical interpretations*. School of Physics Georgia Institute of Technology, Atlanta, GA 30332-0430, USA. http://chaosbook.org/projects/Wen14.pdf

Appendix

Table 4. BIC test using key 0123456789ABCDEF

00112233445566778899aabbccddeeff	**Bit position**	**Standard**	**KSA**	**KSA&PRGA**	**Henon**	**Logistic**
10112233445566778899aabbccddeeff	1st	0.0368	-0.1803	-0.187437	0.172574	-0.265
0**1**112233445566778899aabbccddeeff	2nd	0.3954	-0.0469	0.0134455	0.412945	-0.143017
00**0**1223344556677889 9aabbccddeeff	3rd	0.154994	-0.06426	1	-0.2s15043	-0.257314
001**0**2233445566778899aabbccddeeff	4th	0.1967	-0.01054	-0.177659	-0.00564298	-0.0497381
0011**0**233445566778899aabbccddeeff	5th	-0.3047	-0.04101	0.186674	0.0169692	-0.00876876
00112**0**33445566778899aabbccddeeff	6th	0.244512	0.412698	0.215171	0.183996	0.0566097
001122**2**3445566778899aabbccddeeff	7th	0.407243	0.2639	-0.196862	0.315636	-0.15174
0011223**2**445566778899aabbccddeeff	8th	0.101575	-0.05108	-0.340762	0.137099	-0.518493
00112211344556677889 9aabbccddeeff	7th	0.285129	0.495088	0.0717252	-0.0786881	-0.159492
0011223**1**445566778899aabbccddeeff	8th	0.351805	0.59935	-0.2869	-0.314473	-0.111963
00112233**0**45566778899aabbccddeeff	9th	0.101247	0.5797	-0.168562	0.0490195	-0.135384
0011223340556677889 9aabbccddeeff	10th	-0.236788	-0.054015	-0.453745	0.0710909	0.347558
00112233441566778899aabbccddeeff	11th	-0.24827	-0.1757	0.308554	-0.39449	-0.372436
001122334445667788 99aabbccddeeff	11th	-0.13851	0.21042	-0.0594068	-0.00806681	0.239508
00112233445166778899aabbccddeeff	12th	0.016519	-0.007205	0.0880931	0.138428	0.275291
00112233445466778899aabbccddeeff	12th	-0.0522	-0.412975	-0.1386	-0.0233678	0.413041

(continued)

Table 4. BIC test using key 0123456789ABCDEF (continued)

00112233445566778899aabbccddeeff	**Bit position**	**Standard**	**KSA**	**KSA&PRGA**	**Henon**	**Logistic**
00112233445**2**6778899aabbccddeeff	13th	-0.4539	-0.520097	-0.129466	-0.603409	0.0846676
00112233445**4**6778899aabbccddeeff	13th	0.0812232	-0.019529	0.42662	-0.181944	0.0429341
00112233445562778899aabbccddeeff	14th	-0.0316	0.106366	0.357048	-0.0672107	0.215917
00112233445564778899aabbccddeeff	14th	0.1057	-0.059957	0.037267	-0.31827	0.243821
00112233445566178899aabbccddeeff	15th	-0.035217	0.0697781	-0.133427	-0.644433	-0.219602
00112233445566278899aabbccddeeff	15th	-0.121484	0.52937	0.349139	-0.0387693	-0.245382
00112233445566478899aabbccddeeff	15th	-0.2008	0.447002	-0.0507604	-0.181307	-0.0336706
00112233445566718899aabbccddeeff	16th	-0.25239	0.00703	-0.260094	0.777385	0.129032
00112233445566728899aabbccddeeff	16th	-0.673388	-0.177642	-0.254456	0.0813017	-0.197267
00112233445566748899aabbccddeeff	16th	-0.128997	-0.275698	-0.0786836	0.262353	0.186798
00112233445566770899aabbccddeeff	17th	0.0329197	-0.406862	0.331327	-0.0530885	-0.0552668
00112233445566778099aabbccddeeff	18th	0.0712285	-0.39201	-0.0024714	-0.275308	0.0876923
00112233445566777881**9**aabbccddeeff	19th	0.0975695	0.113024	0.325366	0.335116	-0.25483
00112233445566778889aabbccddeeff	19th	-0.0941015	-0.198437	0.255265	0.0230065	-0.687939
00112233445566778891aabbccddeeff	20th	-0.4177	0.28466	-0.0315297	-0.259197	0.162206
00112233445566778898aabbccddeeff	20th	-0.57307	0.026744	0.569431	0.38584	-0.178433

Table 5. BIC test using key FEDCBA9876543210

00112233445566778899aabbccddeeff	**Standard**	**key only**	**Ksaprga**	**Henon**	**logistic**
10112233445566778899aabbccddeeff	0.417045	-0.220865	-0.57022	0.395484	0.0195112
0**1**112233445566778899aabbccddeeff	-0.434422	0.00901967	-0.159408	0.273016	-0.141773
00**0**122334455667788 99aabbccddeeff	0.0148799	0.346077	-0.311192	0.272182	0.0253394
001**0**223344556677889 9aabbccddeeff	0.326365	0.0452453	-0.099948	0.0555859	-0.216068
0011**0**233445566778899aabbccddeeff	-0.0482196	0.0167129	-0.14916	0.027439	-0.0954986
00112**0**33445566778899aabbccddeeff	-0.216094	0.289786	-0.105732	-0.515892	-0.0669165
001122**2**3445566778899aabbccddeeff	0.0813931	0.172935	0.313931	0.123121	-0.0669165
0011223**2**445566778899aabbccddeeff	-0.23786	0.209435	-0.32757	0.00843778	0.0997132
001122**1**3445566778899aabbccddeeff	0.093428	-0.0669712	-0.139866	0.0591585	0.325533
0011223**1**445566778899aabbccddeeff	0.392115	0.127034	-0.0591007	0.211203	-0.347169
00112233**0**45566778899aabbccddeeff	-0.187959	-0.0109752	0.207049	0.156959	-0.0127717
001122334**0**5566778899aabbccddeeff	-0.165082	0.279662	-0.0133482	0.477197	-0.244642
0011223344**1**566778899aabbccddeeff	0.251234	0.178618	0.12445	-0.202581	0.138681
00112233444566778899aabbccddeeff	0.336015	0.461628	-0.261824	0.322186	-0.106177
001122334451**6**6778899aabbccddeeff	-0.189231	0.0522425	-0.196535	-0.357701	0.0889616
0011223344546**6**778899aabbccddeeff	0.413341	-0.180191	0.0124492	0.164717	0.17522
00112233445526778899aabbccddeeff	-0.0198766	0.0386546	-0.0108255	-0.336237	0.0849657
00112233445546778899aabbccddeeff	0.00733496	-0.104553	0.0703523	0.177754	0.183868
001122334455627778899aabbccddeeff	-0.0305858	0.109162	0.0602006	-0.182223	-0.322914
001122334455647778899aabbccddeeff	-0.191709	-0.273819	0.186209	0.0940754	0.386011
0011223344556617 8899aabbccddeeff	-0.299449	-0.251849	-0.378806	0.351915	-0.146057
0011223344556627 8899aabbccddeeff	-0.0996066	-0.564716	0.120654	-0.0832376	-0.142778
0011223344556647 8899aabbccddeeff	0.0166585	0.0362229	0.0689113	0.516105	0.0995006
0011223344556671 8899aabbccddeeff	0.198812	-0.0355008	-0.411634	-0.0813763	-0.407274
0011223344556672 8899aabbccddeeff	-0.122292	0.267171	0.452606	0.29433	-0.267924
0011223344556674 8899aabbccddeeff	0.20211	0.355754	0.201227	-0.0957974	-0.31905
0011223344556677 0899aabbccddeeff	0.428275	0.0381015	0.0116764	0.481599	0.236903
0011223344556677809 9aabbccddeeff	0.107807	0.0182251	-0.0122187	0.185478	0.348121
0011223344556677881 9aabbccddeeff	-0.164026	0.131867	0.418971	0.363352	0.00416751
0011223344556677888 9aabbccddeeff	-0.209791	0.0444252	-0.282983	-0.616572	0.316093
0011223344556677889 1aabbccddeeff	0.237368	-0.0368011	0.220486	-0.262143	0.334467
0011223344556677889 8aabbccddeeff	-0.207908	-0.108032	-0.101495	0.433086	-0.434455

Table 6. BIC test using Output key F9E8D7C6B5A4012

00112233445566778899aabbccddeeff	**Standard**	**key only**	**Ksaprga**	**Henon**	**Logistic**
10112233445566778899aabbccddeeff	0.417045	0.0585712	0.154629	0.004403	0.185261
0**1**112233445566778899aabbccddeeff	-0.434422	-0.153337	-0.552329	0.0432669	0.0162835
00**0**122334455667788899aabbccddeeff	0.0148799	0.141494	-0.018281	-0.626906	-0.185646
001**0**22334455667788899aabbccddeeff	0.326365	-0.425255	-0.204808	0.0262226	-0.235546
0011**0**2334455667788899aabbccddeeff	-0.0482196	0.00647811	0.254508	-0.0632609	0.274949
00112**0**334455667788899aabbccddeeff	-0.216094	0.176136	-0.240454	-0.0672067	-0.238409
00112**2**2344556677889aabbccddeeff	0.0310626	0.259259	0.127546	0.357445	-0.229124
001122**3**344556677889aabbccddeeff	-0.23786	0.00674504	0.363924	0.528086	0.0444685
001122**1**344556677889aabbccddeeff	0.093428	0.386678	0.471482	-0.227613	0.255397
0011223**1**44556677889aabbccddeeff	0.392115	-0.0881835	-0.36415	0.105209	0.084161
00112233**0**4556677889aabbccddeeff	-0.187959	0.209927	0.388004	-0.16294	0.544963
001122334**0**556677889aabbccddeeff	-0.165082	-0.00630819	-0.00724332	0.000432698	-0.12836
0011223344**1**56677889aabbccddeeff	0.251234	0.449525	-0.00191901	0.0662891	0.062915
0011223344**4**56677889aabbccddeeff	0.336015	0.00630744	-0.0817369	-0.412375	0.353224
00112233445**1**6677889aabbccddeeff	-0.189231	0.21958	-0.248782	-0.415525	0.0225264
00112233445**4**6677889aabbccddeeff	0.413341	-0.0387222	0.577172	0.654266	0.0517412
001122334455**2**677889aabbccddeeff	-0.0198766	0.473408	0.417715	0.105816	-0.178095
001122334455**4**677889aabbccddeeff	0.00733496	0.0179447	-0.0390234	0.425356	-0.42488
0011223344556**2**77889aabbccddeeff	-0.0305858	-0.0362201	-0.484565	-0.00206609	-0.330384
0011223344556**4**77889aabbccddeeff	-0.191709	0.0332508	0.344189	-0.0322695	-0.175936
00112233445566**1**7889aabbccddeeff	-0.299449	-0.229422	-0.391269	-0.235286	-0.316316
00112233445566**2**7889aabbccddeeff	-0.0996066	-0.160023	0.046644	0.27113	-0.160313
00112233445566**4**7889aabbccddeeff	0.0166585	-0.556973	-0.158508	-0.411382	-0.306599
001122334455667**1**889aabbccddeeff	0.198812	0.244194	-0.0908046	-0.170673	-0.292811
001122334455667**2**889aabbccddeeff	-0.122292	0.230191	0.069861	-0.234978	0.12262
001122334455667**4**889aabbccddeeff	0.20211	-0.427662	0.2034	-0.34463	0.426378
00112233445566770899aabbccddeeff	0.428275	0.146083	0.0375816	0.213875	-0.212221
001122334455667780**9**9aabbccddeeff	0.107807	-0.181342	0.411492	0.411069	0.095437
0011223344556677881**9**aabbccddeeff	-0.164026	-0.441504	-0.366232	-0.140391	-0.032834
0011223344556677888**9**aabbccddeeff	-0.209791	-0.066115	0.176382	0.0588114	0.52573
0011223344556677889**1**aabbccddeeff	0.237368	0.168552	-0.674408	0.0176816	0.26591
0011223344556677889**8**aabbccddeeff	-0.207908	-0.302073	-0.22836	-0.254163	0.154455